

ARBEITSBERICHTE DES INSTITUTS FÜR MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG (INFORMATIK)

FRIEDRICH ALEXANDER UNIVERSITÄT ERLANGEN NÜRNBERG

**HERAUSGEBER : H. Billing, W. Händler, U. Herzog,
F. Hofmann, K. Leeb, P. Mertens, H. Müller, G. Nees,
H. Niemann, D. Seitzer, H.J. Schneider**

**Deterministisches Zuordnen in heterogenen
Prozessorsystemen mit Pseudo-Boole Methoden**

Hartmut Wunderatsch

Dissertation

BAND 12 · NUMMER 2 · ERLANGEN · JUNI 1979

1979 # 201



Digitized by the Internet Archive
in 2026

ARBEITSBERICHTE DES INSTITUTS FÜR MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG (INFORMATIK)

FRIEDRICH ALEXANDER UNIVERSITÄT ERLANGEN NÜRNBERG

**HERAUSGEBER : H. Billing, W. Händler, U. Herzog,
F. Hofmann, K. Leeb, P. Mertens, H. Müller, G. Nees,
H. Niemann, D. Seltzer, H.J. Schneider**

Deterministisches Zuordnen in heterogenen Prozessorsystemen mit Pseudo-Boole Methoden

Dem Fachbereich Ingenieurwissenschaften

der Universität Erlangen-Nürnberg

zur Erlangung des Grades

D o k t o r - I n g e n i e u r

vorgelegt

von

Hartmut Wunderatsch

Dissertation

BAND 12 · NUMMER 2 · ERLANGEN · JUNI 1979

ARBEITSBERICHT
DES INSTITUTS FÜR MATHEMATISCHE
MASCHINEN UND DATENVERARBEITUNG
(INFORMATIK)

FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG
VERANTWORTLICHE: H. WILHELM, W. HOFMANN, U. HOFMANN
LEHRSTUHL FÜR: H. WILHELM, W. HOFMANN, U. HOFMANN
LEHRSTUHL FÜR: H. WILHELM, W. HOFMANN, U. HOFMANN
LEHRSTUHL FÜR: H. WILHELM, W. HOFMANN, U. HOFMANN

Deterministisches Zuordnen in heterogenen
Prozessorsystemen mit Pseudo-Boole Methoden



Als Dissertation genehmigt vom
Fachbereich Ingenieurwissenschaften
der Universität Erlangen-Nürnberg

Tag der Einreichung: 12.2.1979

Tag der Promotion: 23. Mai 1979

Dekan: Prof. Dr. D. Seitzer

Berichterstatter: 1. Prof. Dr. F. Hofmann

2. Prof. Dr. K. Leeb

A b s t r a c t

WUNDERATSCH, HARTMUT:

DETERMINISTISCHES ZUORDNEN IN HETEROGENEN PROZESSORSYSTEMEN MIT PSEUDO-BOOLE METHODEN

Es wird gezeigt, daß eine große Klasse deterministischer Zuordnungsprobleme (Schedulingprobleme) in heterogenen Prozessorsystemen mit Pseudo-Boole Methoden lösbar ist. Zwei Fragen werden untersucht: die Existenz von zulässigen Abläufen (Schedules) und das Bestimmen eines optimalen im Fall der Existenz. Ein Zuordnungsproblem wird beschrieben durch ein Arbeitensystem (Tasksystem), das aus zwei Komponenten besteht, dem Arbeitengraph (Taskgraph) und dem Bearbeitungssystem (Prozessorsystem). Der Arbeitengraph legt die Abhängigkeiten zwischen den endlich vielen auszuführenden Aufgaben in Form einer azyklischen Relation fest. Im Bearbeitungssystem werden die Randbedingungen der Durchführung beschrieben. Dort sind Angaben über die Anzahl verfügbarer Prozessoren und teilbarer Betriebsmittel gemacht, sowie aufgabenspezifisch die Rechenzeit auf den heterogenen Prozessoren und die Forderungen nach teilbaren Betriebsmitteln festgehalten. Die Erledigung einer Arbeit darf frühestens in ihrem Freigabezeitpunkt (release date) beginnen und muß spätestens zum Beendigungszeitpunkt (due date) abgeschlossen sein. Eine Kostenfunktion bewertet in Abhängigkeit von den Fertigstellungszeitpunkten aller Aufgaben zulässige Abläufe (Schedules). Unter den alleinigen Voraussetzungen natürlicher Bearbeitungs- und Freigabezeiten und isotoner Kostenfunktionen kann gezeigt werden, daß beide Fragestellungen (Existenz und Optimalität) äquivalent mit der Lösbarkeit Pseudo-Boolescher Gleichungs-/Ungleichungssysteme bzw. Pseudo-Boolescher Optimierungsaufgaben sind und deshalb mit Pseudo-Boole Methoden gelöst werden können. Gerechnete Beispiele zeigen die Brauchbarkeit dieser Technik bei Zuordnungsproblemen.

A b s t r a c t

WUNDERATSCH, HARTMUT:

DETERMINISTIC SCHEDULING IN HETEROGENEOUS PROCESSOR SYSTEMS BY PSEUDO- BOOLEAN METHODS

A Pseudo-Boolean approach to deterministic scheduling in heterogeneous multiprocessor systems is described. Two problems are investigated: the existence of schedules, and, if any exist, the determination of an optimal one. A scheduling problem is defined by a task system, which contains two components, the task graph and the processor system. The task graph describes the precedence constraints of a finite set of tasks by an acyclic relation. The processor system states how many processors and resources exist, how long each of the tasks needs to compute on the different processors, and what resources are needed. The computation for each task can start at its release date and must finish by the due date. A cost function determines the quality of a schedule as a function of all the task finishing times. With the constraints of integer computation times, release dates and isotone cost functions it is shown that both problems (existence and optimality of schedules) are equivalent to the following. Does a solution of a specific pseudo-Boolean equality/inequality system exist, and is it possible to optimize a pseudo-Boolean function under pseudo-Boolean constraints? These questions can be answered by wellknown techniques and lead to useful results in the case of scheduling, as many computed examples show.

Herr Dr. H. J. Schmid hat die Thematik dieser Arbeit mit großem Interesse aufgenommen und mich auf Pseudo-Boole Methoden aufmerksam gemacht. In vielen regen Diskussionen mit ihm, Herrn Prof. Dr. F. Hofmann, Herrn Prof. Dr. K. Leeb und Herrn Prof. Dr. P. L. Hammer wurde die Entwicklung meiner Arbeit gefördert. Dafür allen ein herzliches Dankeschön.

Das gilt ganz besonders auch Frau B. Knechtel, die mit großer Sorgfalt und Hilfsbereitschaft die Reinschrift übernahm und Herrn J. Romig, der alle Zeichnungen hierzu anfertigte.

INHALTSVERZEICHNIS:

	Seite
KAPITEL I: INFORMELLE ERLÄUTERUNGEN DER PROBLEMSTELLUNG	1
I.1 Einführung	1
I.2 Gewählter Zugang zum Zuordnungsproblem (Schedulingproblem)	7
KAPITEL II: FORMALE PROBLEMDEFINITION	15
II.1 Grundlegende Vereinbarung	15
II.2 Definition des Arbeitensystems (Taskssystem)	15
II.3 Ergebnisse zu den Problemen $(G, B, ex, \underline{Q}?)$ und (G, B, opt, μ)	31
KAPITEL III: LÖSBARKEIT VON $(G, B, ex, \underline{Q}?)$ UND (G, B, opt, μ) MIT PSEUDO-BOOLEE METHODEN	39
III.1 Dualität Prozessorfunktion-Arbeitsfunktion	39
III.2 Die Äquivalenz von $(G, B, ex, \underline{Q}?)$ mit der Lösbarkeit eines PB-G/U-Systems (Pseudo-Boolesches Gleichungs- /Ungleichungssystem) [Hammer & Rudeanu 68]	53
III.3 Die Äquivalenz von (G, B, opt, μ) mit der Lösbarkeit einer Pseudo-Boole Optimierungsaufgabe unter Pseudo-Boole Nebenbedingungen [Hammer & Rudeanu 68]	72
KAPITEL IV: ERWEITERUNGEN DER PROBLEMLÖSUNGEN $(G, B, ex, \underline{Q}?)$ UND (G, B, opt, μ)	83
IV.1 Reduktion von Boole-Variablen beim Auftreten linearer Ketten in G	84
IV.2 Nichtunterbrechende Bearbeitung bei natürlichen Rechenzeiten	90
IV.3 Verallgemeinerung eines p-inhomogenen Prozessorsystems	91
IV.4 Berücksichtigung von Verdrängungskosten	103
KAPITEL V: PSEUDO-BOOLEE METHODEN ZUM LÖSEN DER ZUORDNUNGS- PROBLEME $(G, B, ex, \underline{Q}?)$ UND (G, B, opt, μ) UND GE- RECHNETE BEISPIELE	106
V.1 Lineare Pseudo-Boole Methoden	107
V.2 Nicht lineare Pseudo-Boole Methoden	123
V.3 Beispiele für Zuordnungsprobleme, die mit Pseudo-Boole Methoden gelöst wurden	129
V.4 Offene Punkte	142
LITERATUR	145

KAPITEL I: INFORMELLE ERLÄUTERUNGEN DER PROBLEMSTELLUNG

I.1 Einführung

Reihenfolgeprobleme spielen in unserem täglichen Leben eine stets wiederkehrende Rolle. Eine Firma, die termingerecht Aufträge ausführen will, muß auf der Grundlage ihrer Fertigungskapazitäten eine Anordnung wählen, die kostengünstig die damit verbundenen Arbeiten zu erledigen gestattet. Mathematisch betrachtet ist dieses Problem einfach und schwierig zugleich. Einfach deshalb, weil ein Durchprobieren aller möglichen Anordnungen nur endlichen Aufwand bedeutet. Aber genau hierin liegt auch die Schwierigkeit. Der Aufwand ist zwar endlich, aber ungeheuer groß. Bei dieser primitiven Vorgehensweise steigt er wie die Funktion $f(n) = n!$ und würde bereits dann die Rechenleistung sämtlicher auf der Erde installierten Rechenanlagen für Millionen von Jahren benötigen, wenn etwa 100 Aufträge optimal anzuordnen wären. Das liegt daran, daß die Zahl $100!$ größer ist als eine 10 gefolgt von 130 Nullen.

Damit sieht sich der Mathematiker herausgefordert, brauchbare Verfahren zur Lösung dieser Problematik auszudenken und verfügbar zu machen. Das erfordert eine Modellierung der gestellten Aufgabe. Sie wird nachfolgend am Beispiel eines Programms skizziert, das in irgendeiner Programmiersprache formuliert ist und bestmöglich auf einer gegenwärtig diskutierten Rechenanlage ablaufen soll. Ebenso gut ließen sich die im Modell benötigten Eigenschaften an Hand eines Vorgangs aus dem Bereich des Operations Research herausarbeiten.

In heute üblichen Rechnern geschieht die Abarbeitung der Anweisungen eines Benutzerprogramms streng sequentiell. Es macht dabei keinen Unterschied, ob es sich um eine Ein- oder Mehrprozessoranlage handelt, auf der dieses Programm abläuft.

Auch eine betriebssystemseitige Mehrprogrammverarbeitung ändert hieran nichts. Eine Parallelisierung innerhalb der Verarbeitungseinheiten des Rechners, wie dies Befehlspipelining, look-ahead-Mechanismen und dergleichen darstellen, erhöht aus der Sicht des Benutzers zwar die Bearbeitungsgeschwindigkeit, führt jedoch allenfalls zu sehr lokaler Parallelarbeit. Meist wird sie innerhalb einer Anweisung stattfinden und somit für den Anwender unsichtbar bleiben. In Ergänzung hierzu kann das gleichzeitige Arbeiten an mehreren Teilen eines Programms zu weiteren Verkürzungen der Gesamtbearbeitungszeit führen, was bei vielen Problemlösungen ein qualitatives Kriterium ist. Der Wert einer Wettervorhersage zum Beispiel steigt mit ihrer Aktualität. Daß damit tatsächlich eine durchschnittliche Geschwindigkeitssteigerung um den Faktor 10 möglich ist, hat die Untersuchung einer breiten Klasse von existierenden FORTRAN-Programmen durch KUCK et al. [KUCK et al. 74] ergeben. Dabei sind weitere Verbesserungen durch Algorithmen, die speziell auf Parallelverarbeitung zugeschnitten sind, unberücksichtigt geblieben. Eine genauere Betrachtungsweise zeigt nämlich, daß jedes Programm in Teile zerfällt, sogenannte Arbeiten oder Aufgaben (Tasks), deren Erledigung voneinander abhängig sind. Ein Programm könnte etwa die grobe Struktur des nachfolgenden Diagramms besitzen (Bild I.1).

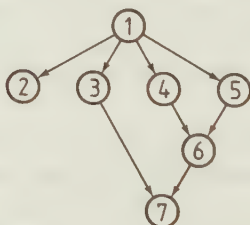


Bild I.1: Arbeitszerlegung eines Beispielprogramms

Es zerfällt in 7 Arbeiten, die teils gemeinsame Daten benötigen, teils aufbereitete Daten weiter verdichten. In der Arbeit 1 werden z.B. die Programmvariablen gesetzt, die in 2 - 5 unabhängig voneinander verwendet werden. Stünden also zur Bearbeitung dieses Programms zwei Prozessoren zur Verfügung, so könnten auch zwei dieser Arbeiten gleichzeitig ausgeführt werden, jedoch erst nach Beenden der Arbeit 1. Die Pfeile im Bild I.1 beschreiben also eine Vorgängerbeziehung zwischen den einzelnen Aufgaben. In diesem Sinn kann 6 erst bearbeitet werden, wenn die Arbeiten 4 und 5 abgeschlossen sind. Analoges gilt für die Aufgabe 7.

Die Gesamtbearbeitungszeit kann mit Hilfe dieser Zerlegung verkürzt werden. Um wieviel, erkennt man bei Konkretisierung des Beispielprogramms. Nehmen wir an, wir hätten zwei gleiche Prozessoren und alle Arbeiten benötigen zur Ausführung ein Einheitsintervall. Dann kann durch geschickte Zuordnung von Arbeiten zu Prozessoren die Gesamtbearbeitung nach 4 Zeitintervallen abgeschlossen sein. Wie dabei vorzugehen ist, zeigt das Gantt-Diagramm von Bild I.2.



Bild I.2 : Gute Zuordnung von Arbeiten zu Prozessoren

Eine ungeschickte Zuordnung von Arbeiten zu Prozessoren hätte die Folge einer um eine Zeiteinheit verlängerten Gesamtausführungszeit (Bild I.3).

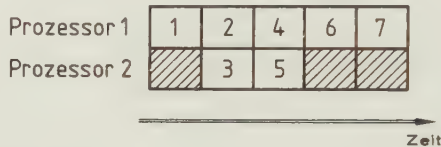


Bild I.3: Schlechte Zuordnung von Arbeiten zu Prozessoren

Zusammenfassend kann festgestellt werden, daß die Gesamtbearbeitungszeit durch Zerlegen eines Programms verkürzbar ist, wenn mehrere Prozessoren verfügbar sind. Die Leistungsfähigkeit der einzelnen Prozessoren braucht dabei nicht erhöht werden. Man überlegt sich leicht Programmbeispiele, in denen bei Vorhandensein hinreichend vieler Prozessoren die Gesamtbearbeitungszeit beliebig verkürzt werden kann. Euphorisch betrachtet lassen sich mit diesem Verfahren unendliche Leistungssteigerungen erzielen, ohne die Hardware von Rechenanlagen schneller machen zu müssen.

Die Zerlegung eines Programms in Aufgaben, die untereinander gewisse Abhängigkeiten besitzen, bietet darüber hinaus weitere Vorteile. Erstens wird dann der Einsatz von Spezialprozessoren sinnvoll, die bestimmte Arbeiten besonders gut ausführen können, und zweitens können dadurch teilbare Betriebsmittel feiner gestuft vergeben werden, was zu deren effektiveren Auslastung führt.

Während bei sequentieller Abarbeitung eines Programms der Einsatz von Spezialprozessoren zur Bearbeitung bestimmter Programmbereiche exklusiv zu Tätigkeiten von Standardprozessoren erfolgt, könnten nach einer Zerlegung in teilweise unabhängige Arbeiten beide Prozessortypen parallel mit verschiedenen Aufgaben des gleichen Programms beschäftigt sein. Als Spezialprozessoren sind Gleitkomma-, Textverarbeitungs-, Array-Prozessoren usw. denkbar, die z.B. Gleitkomma-Arithmetik, Byte-Manipulationen und Matrix-Operationen besonders gut beherrschen.

Nun sind zur Bearbeitung eines Programms bekannterweise nicht nur Prozessoren erforderlich, sondern auch weitere Betriebsmittel, die sogenannten teilbaren Betriebsmittel. Teilbar deshalb, weil sie gleichzeitig von mehreren Programmen in Anspruch genommen werden können. Im einzelnen handelt es sich um die Betriebsmittel Arbeitsspeicher, Hintergrundspeicher, Bandgeräte, sonstige E/A-Geräte udgl., die an einer Rechenanlage mehrfach vorhanden sind und deshalb, als Gesamtheit betrachtet, gleichzeitig von mehreren Programmen belegt sein können. Während bei sequentieller Bearbeitung diese Betriebsmittel programmglobally vergeben werden müssen, können sie nun aufgabenspezifisch zugeordnet werden und sind somit effektiver nutzbar.

Arbeiten können also solange parallel ausgeführt werden, wie genügend Prozessoren und teilbare Betriebsmittel vorhanden sind. Die Forderungen nach letzteren sind im vorangehenden Beispiel (Bild I.1) außer acht gelassen. Die Aufgaben 4 und 5 könnten zum Beispiel nicht mehr parallel ausgeführt werden, wenn diese gemeinsam 3 Bandgeräte benötigen, aber nur 2 an der Rechenanlage vorhanden sind.

Bei Programmen, die Prozeßsteuerungen vornehmen, kommt eine weitere Einschränkung während des Programmlaufs hinzu: es muß zeitgerecht reagiert werden. So darf z.B. ein Meßwert erst dann verarbeitet werden, wenn das zugehörige Meßgerät im eingeschwungenen Zustand ist. Aber es muß auch sichergestellt sein, daß Programmteile, die im Alarmfall ein Fließband abstellen, rechtzeitig ausgeführt sind. So rechtzeitig z.B., daß ein Werkstück eine Montagestation noch nicht passiert hat, an der es noch verändert werden muß: Die Durchführung gewisser Arbeiten muß also innerhalb von Zeitschranken erfolgen, die von "außen" vorgegeben sind. Bei sequentieller Bearbeitung solcher Programme können die Zeitvorgaben nur dadurch eingehalten werden, daß der Programmierer die steuernden Programmanweisungen an bestimmten Stellen seines Programms einfügt und durch Zeitabfragen sichert. Liegt jedoch eine Arbeitenzerlegung vor, so kann einfach gefordert werden, daß prozeßsteuernde Teile innerhalb vorgegebener Zeitschranken ausgeführt werden. Eine Zuordnung der Arbeiten zu Prozessoren muß in einem vom eigentlichen Programm getrennten Schritt so bestimmt werden, daß neben der Beschränkung durch die Aufgabenabhängigkeiten und den Forderungen nach teilbaren Betriebsmitteln zusätzlich die Arbeiten auch noch innerhalb vorgegebener Zeitschranken erledigt werden.

Damit stellen sich zwei Fragen. Gibt es überhaupt Zuordnungen von Aufgaben zu Prozessoren, im folgenden Ablaufpläne oder Abläufe (Schedules) genannt, die alle Randbedingungen einhalten? Falls ja, wie sieht dann ein "optimaler" Ablaufplan aus? Optimal kann in diesem Zusammenhang bedeuten, daß es sich um denjenigen handelt, der die kürzest mögliche Gesamtbearbeitung des Programms sicherstellt oder allgemein bezüglich einer vorgegebenen Kostenfunktion minimal ist.

In vielen Fällen wird diese von der Durchführung jeder einzelnen Arbeit abhängig sein:

von deren Fertigstellungszeitpunkt,
von ihrer Bedeutung innerhalb des Gesamtprogramms,
von der Anzahl der Unterbrechungen während ihrer
Erledigung, usw..

Beide Fragestellungen können unter verschiedenen Gesichtspunkten präzisiert werden. Eine probabilistische und deterministische Betrachtungsweise ist möglich.

I.2 Gewählter Zugang zum Zuordnungsproblem (Schedulingproblem)

Der probabilistische Ansatz berücksichtigt, daß benötigte Rechenzeiten für die Durchführung der Arbeiten nicht immer von vorneherein bekannt sind, sondern allenfalls deren Verteilungsfunktion. Demzufolge kann auch nur eine Wahrscheinlichkeitsaussage über die Gesamtbearbeitungszeit bzw. die Kosten bestimmter Ablaufpläne erzielt werden. Dieser Zugang wird umso komplexer, je differenzierter der Ankunftsprozeß beschrieben wird. Neben der Rechenzeitverteilung der Arbeiten können weitere statistische Annahmen über Betriebsmittelforderungen, Arbeitsabhängigkeiten und Bearbeitungsintervalle gemacht werden, um Ergebnisse über verkehrstheoretische Ansätze zu erzielen. Die exponentielle Verteilung spielt hierbei eine zentrale Rolle. Einen ersten Überblick über Arbeiten mit solchen Zielsetzungen gibt [Coffman & Denning 73].

In dieser Arbeit werden die Zuordnungsprobleme (Schedulingprobleme) deterministisch betrachtet. Die exakten Werte der Bearbeitungszeiten, Arbeitsabhängigkeiten (Taskstruktur), Betriebsmittelforderungen, Freigabe- und Beendigungszeitpunkte sind von Anfang an bekannt. Die Erledigung einer Aufgabe kann dann innerhalb eines vorgegebenen Zeitintervalls unter Berücksichtigung der vorgegebenen Reihenfolge und der vorhandenen Betriebsmittel stattfinden.

Und zwar grundsätzlich in zwei Arten: nichtunterbrechend (nonpreemptive) und unterbrechend (preemptive). Während im ersten Fall eine Arbeit nach Zuteilung zu einem Prozessor diesen solange belegt, bis sie zu Ende gerechnet ist, sind im zweiten Fall abschnittsweise Bearbeitungen möglich. Ein Prozessor übernimmt eine Arbeit, führt sie jedoch nicht vollständig aus, sondern "verdrängt" sie vor ihrem Fertigstellungszeitpunkt. Zu einem späteren Zeitpunkt kann deren Bearbeitung auf einem geeigneten Prozessor fortgesetzt werden, der nicht unbedingt mit dem erstbearbeitenden identisch sein muß. Insgesamt sind beliebig, aber endlich viele Unterbrechungen zulässig. Aufgrund der Tatsache, daß jede nichtunterbrechende Bearbeitung trivialerweise eine unterbrechende Bearbeitung ist (nämlich eine mit 0 Verdrängungen), liegt die Vermutung nahe, daß unterbrechende Arbeitsabläufe eher existieren und günstigere Kosten verursachen. Dies bestätigen bereits einfachste Beispiele. Sind z.B. drei voneinander unabhängige Arbeiten zu erledigen, die alle die gleiche Bearbeitungszeit auf jedem von zwei vorhandenen Prozessoren benötigen und die Bearbeitungsrandbedingungen überdies weder durch Forderungen nach teilbaren Betriebsmitteln noch durch Bearbeitungsintervalle eingeschränkt, so benötigt der kürzeste Gesamtablauf im unterbrechenden Fall 1,5 Zeiteinheiten bei insgesamt 2 Unterbrechungen, im nichtunterbrechenden Fall jedoch 2 Zeiteinheiten. Die Gantt-Diagramme im Bild I.4 veranschaulichen das.

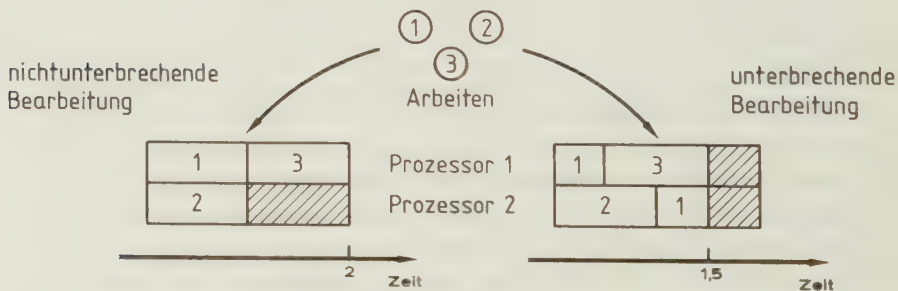


Bild I.4: Unterschied von unterbrechender und nichtunterbrechender Bearbeitung

Die in diesem Sinn günstigere Bearbeitung bei unterbrechender Strategie berücksichtigt jedoch den Aufwand für Verdrängungen nicht. Die meisten Arbeiten, die sich im Bereich des deterministischen Zuordnens mit unterbrechender Bearbeitung auseinandersetzen, gehen davon aus. Bei realen Prozessorsystemen kann dies jedoch nicht getan werden, denn längst ist bekannt, wie empfindlich allzuviele Verdrängungen das Ablaufgeschehen in einer Rechenanlage stören. Aus diesem Grund scheint es sinnvoll, Unterbrechungen nur zu ganz bestimmten Zeitpunkten zuzulassen und deren Kosten zu berücksichtigen. Hierdurch ist automatisch eine obere Schranke der maximal stattfindenden Verdrängungen gegeben. Diesem Sachverhalt wird in der vorliegenden Arbeit Rechnung getragen.

Neben der Beschränkung auf deterministische Abläufe werden ausschließlich nichtunterbrechende Ablaufpläne betrachtet oder solche, bei denen Unterbrechungen allenfalls zu vorher fixierten Zeitpunkten stattfinden dürfen. Es wird gezeigt, daß es der hier gewählte Ansatz zuläßt, beide Betrachtungsweisen auseinander abzuleiten.

Alle Ergebnisse über das Finden deterministischer Ablaufpläne stehen im Spannungsfeld zweier Extremaleigenschaften. Zum einen ist bekannt, daß das Bestimmen nichtunterbrechender Abläufe ein endliches Problem ist und in diesem Sinn natürlich trivial. Das gleiche gilt auch für Ablaufpläne, die Verdrängungen zu festgesetzten Zeitpunkten zulassen. Zum anderen kennt man deren allgemeine NP-Vollständigkeit, die sich bereits bei scheinbar einfachen Zuordnungsproblemen zeigt. Dies gilt etwa für die Bestimmung des kürzesten Ablaufplanes für unabhängige Arbeiten, die ein teilbares Betriebsmittel benötigten und alle auf drei identischen Prozessoren eine Bearbeitungseinheit in Anspruch nehmen [Garey & Johnson 75].

Die wesentliche Eigenschaft NP-vollständiger Probleme besteht darin, daß sie im folgenden Sinn äquivalent sind: ist eines dieser Probleme durch einen Algorithmus lösbar, der polynomial beschränkt viele Schnitte braucht, so gilt dies auch für alle anderen NP-vollständigen Probleme [Valiant 78]. Für das oben zitierte spezielle 3-Prozessor-Zuordnungsproblem einen Algorithmus zu finden, der in höchstens $c \cdot n^k$ Schritten den kürzesten Ablaufplan erstellt, wobei c und k probleminvariante Größen sind und n die Anzahl der zu erledigenden Arbeiten beschreibt, ist genauso schwer wie das polynomial beschränkte Lösen des allgemeinen Zuordnungsproblems, das mehr als 3 Prozessoren zuläßt und z.B. beliebige Vorgängerbeziehungen.

Aufgrund der Endlichkeit kann man sinnvollerweise ein Verfahren nicht gut nennen, das alle möglichen Ablaufpläne untersucht, um das gestellte Problem zu lösen. Es sind nämlich exponentiell viele, da Ablaufpläne in zahlreichen Fällen durch eine Permutation der n Arbeiten - hiervon gibt es $n!$ - beschrieben werden können. Um selbst bei kleinem n alle Permutationen z.B. auf Optimalität überprüfen zu können, sind auf heute leistungsstark zu nennenden Rechenanlagen Zeiten nötig, die außerhalb der mittleren Fehlerrate solcher Maschinen liegen und damit praktisch nicht berechenbar sind. Im Gegensatz hierzu gibt es die Übereinstimmung, daß Algorithmen, die in Abhängigkeit von der Problemgröße n (etwa die Anzahl von Arbeiten bei Zuordnungsproblemen) nach maximal cn^k (c und k sind unabhängig von n) Schritten eine Lösung gefunden haben, auf heutigen Rechenanlagen in annehmbarer Zeit zu Ergebnissen führen. Aus diesem Grund wurden im Bereich des deterministischen Zuordnens von Anfang an Algorithmen gesucht, die eine gegebene Problemklasse polynomial beschränkt lösen. Andere werden teilweise für nicht zulässig erklärt [Ecker 77].

Eine derartige Beurteilung ist dennoch aus mehreren Gründen praxisfremd. Zwei Gründe seien hier genannt. Zum einen ist ein solches Komplexitätsmaß auf den schlechtest möglichen Fall abgestimmt. D.h. wenige "pathologische" Fälle in einer Problemklasse können einen Algorithmus als schlecht klassifizieren, obwohl er in den meisten Fällen gute Laufzeiten besitzt. Dem könnte dadurch abgeholfen werden, daß das Gütemaß gewichtet über alle Probleme einer Problemklasse gemittelt wird. Die Gewichte sind dann z.B. durch Wahrscheinlichkeitsverteilungen der Probleme selbst gegeben. Zum anderen ist es ein Untersuchungsergebnis gerade im Bereich des deterministischen Zuordnens, das die Praxisferne einer derartigen Algorithmenbeurteilung aufzeigt. Bei homogenen Prozessorsystemen - jede Arbeit kann von jedem Prozessor gleich gut erledigt werden - weicht z.B. der schlechteste vom besten Ablaufplan bezüglich der Gesamtbearbeitungszeit maximal um den Faktor 2 ab. Bezieht man nun die Zeit, die ein Algorithmus zur Bestimmung der optimalen Zuordnung benötigt, in die Ausführungszeit des erstellten Ablaufplans ein, so war seine Bestimmung nur sinnvoll, wenn die Gesamtzeit kürzer ist als die Ausführung des schlechtesten Ablaufplans. Ansonsten wäre die Ausführungszeit einer beliebig gewählten Zuordnung in jedem Fall günstiger gewesen. Der Gebrauch eines polynomial beschränkten Algorithmus verhindert das nicht.

Umgekehrt kann aber auch das Anwenden eines nicht nachweislich polynomial beschränkten Zuordnungsalgorithmus', der z.B. optimale Ablaufpläne berechnet, sinnvoll sein. Dann nämlich, wenn er wenigstens für eine Teilklasse des Problems in der Praxis zu akzeptablen Rechenzeiten führt, die wettgemacht werden durch die erzielte minimale Verarbeitungszeit. Gerade bei Produktionsläufen, bei Programmen also, die häufig ausgeführt werden, kann sich eine solche Vorgehensweise schnell ausgeglichen haben. Natürlich wird man zu diesen Zuordnungsalgorithmen, die auch etwas unglücklich abzählende (enumerative) Algorithmen genannt werden, erst dann greifen, wenn polynomial beschränkte für das gegebene Problem nicht existieren.

Da Algorithmen, insbesondere polynomial beschränkte, nur für stark eingeschränkte Probleme bekannt sind, hat man mit dem Pseudo-Boole Ansatz wenigstens ein Mittel an der Hand, um überhaupt rechnen zu können.

Die Verwendung abzählender Algorithmen kann aber geeigneter sein, als der Rückzug auf vorhandene heuristische Lösungen, bei denen allenfalls Schranken für die maximale Abweichung zu erzeugten optimalen Ablaufplänen vorhanden sind. Die Entscheidung zwischen der praktischen Verwendung von heuristischen und nicht nachweislich polynomial beschränkten Algorithmen wird demzufolge von zwei wichtigen Einflußgrößen abhängen:

1. Von der Güte des heuristischen Algorithmus'; seine Güte muß mit Null veranschlagt werden, wenn keine Abschätzung zu optimalen Ablaufplänen existiert;
2. von der Anzahl der Programmläufe, die mit dem erzeugten Ablaufplan durchgeführt werden.

Leider steht bei vielen Zuordnungsproblemen diese Entscheidung gar nicht erst zur Diskussion, da heuristische Algorithmen mit einem nicht trivialen Gütemaß nur in wenigen Fällen bekannt sind [Graham et al. 77]. Der Bereich nicht verschwindender Betriebsmittelanforderungen und das Einbeziehen von strikten Bearbeitungsintervallen (deadlines) entzieht sich ebenso solchen Ergebnissen, wie dies Zuordnungsprobleme in inhomogenen Prozessorsystemen tun. So gibt es z.B. keine Verfahren, die Gesamtzeit-optimale Ablaufpläne für beliebige Arbeitsabhängigkeiten und mehr als zwei identische Prozessoren erstellen, selbst wenn Einheitsbearbeitungszeiten, triviale Betriebsmittelforderungen und Freigabe- und Beendigungszeitpunkte vorausgesetzt werden.

Im Bereich inhomogener Prozessorsysteme sind Ergebnisse so gut wie nicht vorhanden [Graham 77]. Zusammenfassend kann festgestellt werden, daß komplexe Zuordnungsprobleme bislang nicht gelöst sind. Im Anschluß an die formale Problemdefinition werden die bisher erzielten Ergebnisse aufgeführt. Sie zeigen, wie eingeschränkt diese Einzellösungen anwendbar sind. Eine Betrachtung der mathematischen Struktur solcher Zuordnungsprobleme ist bis zum heutigen Zeitpunkt übergreifend nicht erfolgt.

In dieser Arbeit werden deterministische, nichtunterbrechende Zuordnungsprobleme bezüglich ihrer mathematischen Struktur untersucht und mit Methoden der Analysis charakterisiert. Unter der Einschränkung rationaler Bearbeitungszeiten und Freigabezeitpunkte kann hieraus eine algebraische Äquivalenz hergeleitet werden, die mit Pseudo-Boole Methoden [Hammer & Rudeanu 68] behandelt werden kann und konstruktiv Zuordnungsalgorithmen (Schedule-Algorithmen) liefert, die folgende Fragestellungen lösen:

1. existiert zu einem gestellten Zuordnungsproblem überhaupt ein Ablaufplan?
2. falls ja, wie sieht ein kostenoptimaler Ablaufplan aus?

Die Ergebnisse werden erzielt für homogene und heterogene Prozessorsysteme allgemeinsten Art mit beliebiger Anzahl von Prozessoren, bei beliebigen Arbeitsabhängigkeiten, bei beliebigen Forderungen nach teilbaren Betriebsmitteln, bei beliebigen Beendigungszeitpunkten (deadlines). Lediglich zwei Voraussetzungen müssen erfüllt sein. Die Bearbeitungszeiten und die Arbeitsfreigabezeitpunkte müssen rational sein und die Kostenfunktion isoton.

Beide Bedingungen sind in der Praxis erfüllt, insbesondere trifft die zweite bei den gängigen Kostenfunktionen "Gesamtbearbeitungszeit", "mittlere Flußzeit" und "mittlere gewichtete Flußzeit" zu. Die unter diesen Voraussetzungen hergeleitete algebraische Äquivalenz kann im Sinne Dijkstra's als nicht determinierter Prädikamentransformer verstanden werden [Dijkstra 76], dessen Determinierung alle möglichen Zuordnungsalgorithmen liefert. Es ist mathematisch klar, daß ein Zuordnungsalgorithmus, der Ablaufpläne erstellt, die die gefundene algebraische Äquivalenz erfüllen, sogar kostenoptimale bestimmt. Korrektheitsbeweise können somit für eine große Klasse von Zuordnungsproblemen über diesen Prädikamentransformer [Steinacker 78] durchgeführt werden. Die Klasse umfaßt NP-vollständige Fragestellungen, weswegen bisher keine Abschätzung irgendeines Komplexitätsmaßes der so abgeleiteten Zuordnungsalgorithmen gemacht wurde. Die Anwendung auf eine Reihe teilweise auch umfangreicher Beispiele zeigt jedoch, daß sich die Anzahl der benötigten Schritte als so gering erwies, daß sie auf heute üblichen Rechenanlagen (sequentielle Programmbearbeitung) im Sekundenbereich ausgeführt werden können. Da überdies die verwendeten Pseudo-Boole Methoden hochgradig parallelisierbar sind und auch das Interesse der Mathematik an parallelen Algorithmen zunehmend steigt [Heller 78], werden die in dieser Arbeit angewandten Techniken weitere interessante Ergebnisse erwarten lassen.

KAPITEL II: FORMALE PROBLEMDEFINITION

II.1 Grundlegende Vereinbarungen

Kleinbuchstaben des griechischen Alphabetes beschreiben in allen folgenden Kapiteln Abbildungen, Großbuchstaben des gleichen Alphabetes Mengen von Abbildungen. In Ergänzung hierzu werden Groß- und Kleinbuchstaben des lateinischen Zeichensatzes für sonstige Mengen und deren Elemente benützt. Unterstrichene Symbole kennzeichnen einen Vektor, dessen Komponenten durch das indizierte gleiche Symbol dargestellt werden, z.B.

$$\underline{\tau} = (\tau_1, \tau_2, \dots, \tau_m).$$

II.2 Definition des Arbeitensystems (Tasksystems)

Ein Arbeitensystem umfaßt zwei Bestandteile, den Arbeitsgraph (Taskgraph) und das Bearbeitungssystem.

Def. A (Kap. II.2): Eine Tupel $G = (T, <)$ heißt Arbeitsgraph (Taskgraph), wenn

- (i) $T = \{a_i : 1 \leq i \leq n\}$ mit $n \in \mathbb{N}$
- (ii) $< \subseteq T^2$ gerichtete, azyklische Relation auf T

Die Elemente von T heißen Arbeiten oder Aufgaben (Tasks) a_i ($1 \leq i \leq n$). Die Relation $<$ beschreibt die Vorgängerbeziehung innerhalb der Arbeitsmenge T .

Gilt $a_i < a_j$ für $1 \leq i, j \leq n$, so wird das so verstanden, daß die Arbeit a_i ausgeführt sein muß, bevor die Bearbeitung von a_j begonnen werden kann.

Mit dieser Interpretation der Vorgängerbeziehung $<$ ist klar, daß man sich auf Erzeugendensysteme beschränken kann, d.h. auf Relationen mit der Eigenschaft:

$$\begin{array}{c} \wedge \\ 1 \leq i, j, k \leq n \end{array} \quad (a_i < a_j < a_k \Rightarrow a_i < a_k)$$

Die Forderung nach azyklischen Vorgängerbeziehungen zwischen den Arbeiten scheint auf den ersten Blick eine Einschränkung gegenüber wirklichen Programmen zu sein, die Schleifen beinhalten. Sie kann jedoch so begründet werden: Zyklen in einem Arbeitsgraph können zu einer einzigen Arbeit zusammengefaßt werden, wodurch schließlich ein zyklensfreier Graph entsteht [Ramamoorthy et al. 72]. Überdies ist das Zuordnen (Scheduling) ständig wiederkehrender Anforderungen Gegenstand eigener Untersuchungen [Schindler 75]. Eine weitere Begründung für eine azyklische Relation liefert die Forderung nach strukturierter Programmierung. Die Arbeiten werden dann nicht nur nach dem Gesichtspunkt maximaler Parallelisierbarkeit ausgesucht, sondern auch als Moduln verstanden werden müssen, die untereinander in hierarchischer Beziehung stehen. Diese Hierarchie verbietet Zyklen.

Def. B (Kap. II.2): Sei $G = (T, <)$ ein Arbeitsgraph (Task-graph) und $a_i \in T$. Dann heißen:

- (i) $<^+$ die transitive Hülle von $<$,
- (ii) $UV(a_i) = \{a_j \in T : a_j < a_i\}$ die Menge aller unmittelbaren Vorgänger von a_i ,
- (iii) $UN(a_i) = \{a_j \in T : a_i < a_j\}$ die Menge aller unmittelbaren Nachfolger von a_i ,
- (iv) $V(a_i) = \{a_j \in T : a_j <^+ a_i\}$ die Menge aller Vorgänger von a_i ,
- (v) $N(a_i) = \{a_j \in T : a_i <^+ a_j\}$ die Menge aller Nachfolger von a_i .

Eine Arbeit $a \in T$ heißt Startarbeit, falls $UV(a) = \emptyset$, sie heißt Endarbeit, falls $UN(a) = \emptyset$.

Es ist trivial, daß zwischen der Menge der unmittelbaren Vorgänger und Nachfolger und der Menge der Vorgänger und Nachfolger einer Arbeit $a \in T$ gilt:

$$(*) \quad UV(a) = \emptyset \Leftrightarrow V(a) = \emptyset$$

$$(**) \quad UN(a) = \emptyset \Leftrightarrow N(a) = \emptyset$$

Die Beschreibung von Zuordnungsproblemen erfordert über den Arbeitengraph (Taskgraph) hinaus Angaben über deren Rechenzeitbedarf, über Forderungen nach teilbaren Betriebsmitteln, über Zeitintervalle, innerhalb derer die Bearbeitung stattfinden soll und über eine zugrundegelegte Kostenfunktion. Diese Angaben werden im sog. Bearbeitungssystem gemacht, das zusätzlich die Anzahl und den Typ vorhandener Prozessoren und Betriebsmittel festhält.

Def. C (Kap. II.2): Sei $G = (T, <)$ ein Arbeitengraph mit $n \in \mathbb{N}$ Arbeiten. Ein Quintupel $B = (\underline{\tau}, \underline{\mu}, \tau_f, \tau_e, \kappa)$ heißt Bearbeitungssystem mit $m \in \mathbb{N}$ Prozessoren und $s \in \mathbb{N}$ teilbaren Betriebsmitteln (kurz: Bearbeitungssystem), falls:

$$(i) \quad \underline{\tau} : T \rightarrow (\mathbb{R}^+)^m \quad \mathbb{R}^+ = \{r \in \mathbb{R} : r > 0\} \cup \{\infty\}$$

$$a \rightsquigarrow (\tau_1(a), \tau_2(a), \dots, \tau_m(a))$$

hierbei muß gelten: $\tau_1(a) \neq \infty$
 $a \in T \quad 1 \leq i \leq m$

$$(ii) \quad \underline{\mu} : T \rightarrow [0, 1]^s$$

$$a \rightsquigarrow (\mu_1(a), \mu_2(a), \dots, \mu_s(a))$$

$$(iii) \quad \tau_f : T \rightarrow \mathbb{R}^{\geq 0} \quad \mathbb{R}^{\geq 0} = \{r \in \mathbb{R} : r \geq 0\}$$

$$a \rightsquigarrow \tau_f(a)$$

$$(iv) \quad \tau_e : T \rightarrow \mathbb{R}^{> 0} \quad \mathbb{R}^{> 0} = \{r \in \mathbb{R} : r > 0\}$$

$$a \rightsquigarrow \tau_e(a)$$

$$(v) \quad \kappa : (\mathbb{R}^{> 0})^n \rightarrow \mathbb{R}$$

$$(x_1, x_2, \dots, x_n) \rightsquigarrow \kappa(x_1, x_2, \dots, x_n)$$

Die Abbildung τ beschreibt den Rechenzeitbedarf jeder Aufgabe auf allen m Prozessoren. Sie wird Rechenzeit genannt. Für $1 \leq i \leq m$ gibt $\tau_i(a)$ die Bearbeitungsdauer der Arbeit $a \in T$ auf dem Prozessor i an. Es ist sichergestellt, daß sie auf mindestens einem Prozessor in endlicher Zeit fertig werden kann (siehe (i)). Gilt für ein $1 \leq i \leq m$ $\tau_i(a) = \infty$, dann wird das so gedeutet, daß Aufgabe a auf dem Prozessor i nicht bearbeitbar ist.

Aus dieser allgemeinen Definition des sogenannten heterogenen Prozessorsystems lassen sich zwei Sonderfälle ableiten: das homogene und das inhomogene. Gelegentlich ist in der Literatur auch die Rede von einem uniformen Prozessorsystem. In solchen besitzen die Prozessoren untereinander feste Leistungsfaktoren, es gilt:

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq i \leq m \end{array} \quad \begin{array}{c} \diagdown \quad \diagup \\ c_i \in \mathbb{R}^{>0} \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ a \in T \end{array} \quad \tau_i(a) = c_i \tau_1(a)$$

Def. D (Kap. II.2): Sei $G = (T, <)$ ein Arbeitengraph mit dem Bearbeitungssystem $B = (\tau, \mu, \tau_f, \tau_e, \kappa)$. B besitzt ein homogenes Prozessorsystem, falls die Abbildung τ folgenden Bedingungen genügt:

$$(i) \quad \begin{array}{c} \diagup \quad \diagdown \\ a \in T \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq i, j \leq m \end{array} \quad \tau_i(a) = \tau_j(a)$$

Das Bearbeitungssystem verfügt über ein p-inhomogenes Prozessorsystem ($p \in \mathbb{N}$), falls eine Folge

$$0 = m_0 < m_1 < m_2 < \dots < m_p = m$$

natürlicher Zahlen m_i ($1 \leq i \leq p$) existiert und eine Partition

$$T = \bigcup_{i=1}^p T^{(i)}$$

der Arbeitsmenge mit den Eigenschaften

$$(i)^* \quad \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq i \leq p \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ a \in T^{(i)} \end{array} \quad (m_{i-1} < j, k \leq m_i \Rightarrow \tau_j(a) = \tau_k(a) \neq \infty)$$

$$(ii)^* \quad \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq i \leq p \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ a \in T^{(i)} \end{array} \quad (j \leq m_{i-1} \vee j > m_i \Rightarrow \tau_j(a) = \infty)$$

In einem homogenen Prozessorsystem ist es gleichgültig, auf welchem Prozessor eine Arbeit ausgeführt wird. Ihr Rechenzeitbedarf ist überall gleich groß. Anders ausgedrückt: alle Prozessoren sind vom gleichen Typ.

Demgegenüber steht ein p-inhomogenes Prozessorsystem aus p verschiedenen Prozessortypen. Vom Typ 1 gibt es $(m_1 - m_0)$ viele, vom Typ 2 $(m_2 - m_1)$ Stück, allgemein vom Typ i $(1 \leq i \leq p)$ $(m_i - m_{i-1})$ Prozessoren. Das Erledigen einer Arbeit a kann stets nur durch Prozessoren eines einzigen Typs vorgenommen werden. Ist $a \in T^{(i)}$, so kommt ausschließlich der Prozessortyp i in Frage (Eigenschaft (i)*). Auf allen anderen Prozessoren ist diese Arbeit nicht durchführbar (Eigenschaft (ii)*).

Das Bild II.1 stellt die allgemeine Rechenzeit τ den beiden Sonderfällen homogenes und p-inhomogenes Prozessorsystem gegenüber.

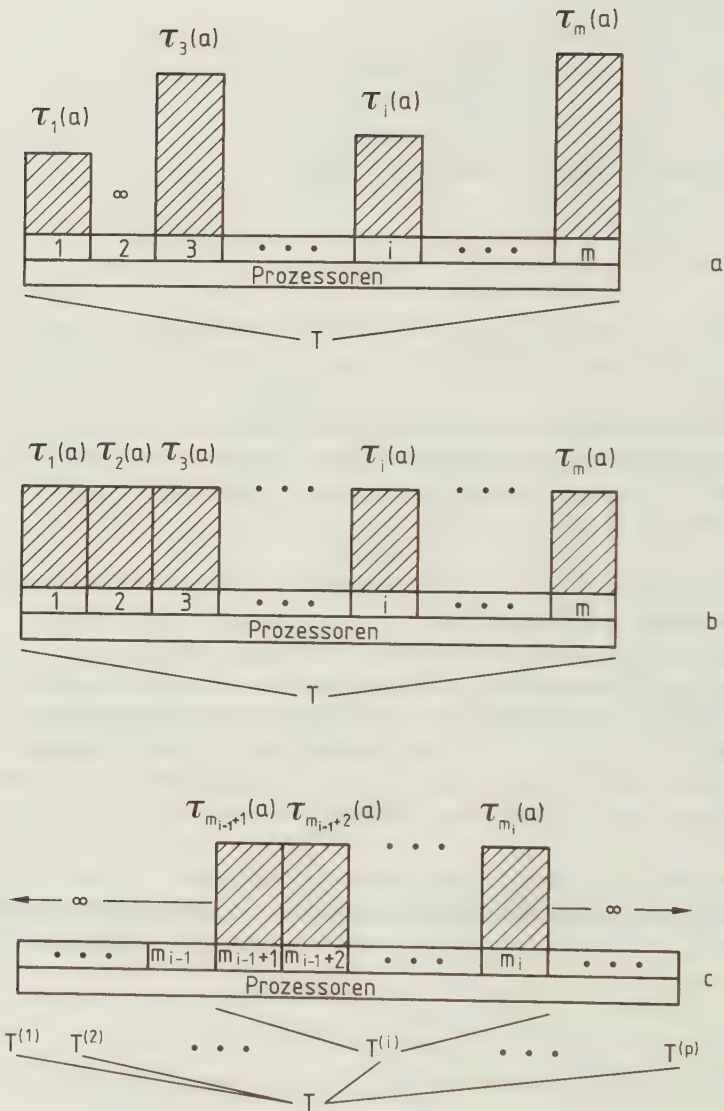


Bild II.1: Rechenzeit einer Arbeit $a \in T$ bzw. $T^{(i)}$

a: Allgemeine Form

b: Homogenes Prozessorsystem

c: Inhomogenes Prozessorsystem

Die Abbildung $\underline{\mu}$ von Def. C (Kap. II.2) charakterisiert arbeitenspezifisch die Forderungen nach teilbaren Betriebsmitteln. Sie heißt Betriebsmittelbedarf der Arbeiten.

$\mu_i(a)$ ($1 \leq i \leq s$) beschreibt den prozentualen Anteil des teilbaren Betriebsmittels i , das zur Erledigung der Arbeit $a \in T$ benötigt wird. Insgesamt stehen $s \in \mathbb{N}$ verschiedene Betriebsmittel zur Verfügung.

Der Freigabezeitpunkt, also der Zeitpunkt, ab dem die Durchführung einer Arbeit $a \in T$ frühestens begonnen werden kann, ist durch den Funktionswert $\tau_f(a)$ festgelegt. τ_f heißt Freigabefunktion. Analog hierzu legt die Abbildung τ_e den Zeitpunkt der spätesten Beendigung jeder Arbeit fest. τ_e heißt Beendigungsfunktion.

Es ist klar, daß ein Ablaufplan für die Erledigung der anstehenden Arbeiten - was das genau ist, sehen wir noch - überhaupt nur existieren kann, wenn gilt:

$$\bigwedge_{a \in T} \min \{ \tau_i(a) : 1 \leq i \leq m \} \leq \tau_e(a) - \tau_f(a)$$

Das heißt, das einzuhaltende Durchführungsintervall muß für jede Arbeit so groß sein, daß mindestens ein geeigneter Prozessor damit auskommt.

Die Funktion κ ermittelt in Abhängigkeit von den Fertigstellungszeitpunkten x_i der einzelnen Arbeiten a_i Kosten für das Erledigen aller Arbeiten gemäß einem bestimmten Ablaufplan. Da verschiedene Ablaufpläne unterschiedliche Fertigstellungszeitpunkte der Aufgaben nach sich ziehen, kann über κ eine Bewertung vorgenommen werden, die es schließlich erlaubt, von optimalen Ablaufplänen (optimalen Schedules) zu sprechen.

Bevor dies jedoch exakt formuliert werden kann, sind weitere Definitionen nötig. Typische Kostenfunktionen sind die folgenden:

$$(i) \quad \kappa(x_1, x_2, \dots, x_n) = x_{i_0} \quad \text{für ein } 1 \leq i_0 \leq n$$

$$(ii) \quad \kappa(x_1, x_2, \dots, x_n) = \frac{1}{n} \sum_{i=1}^n x_i$$

$$(iii) \quad \kappa(x_1, x_2, \dots, x_n) = \frac{1}{n} \sum_{i=1}^n g(a_i) x_i \text{ mit } g(a_i) \in \mathbb{R} \text{ für alle } 1 \leq i \leq n$$

Es wird sich herausstellen, daß es diese Kostenfunktionen sind, die die Gesamtbearbeitungszeit, den mittleren ungewichteten und gewichteten Fluß eines Arbeitengraphen beschreiben. Das Einbeziehen von Unterbrechungen in die Kostenberechnung ist im Moment außer acht gelassen. In einem späteren Kapitel wird diese Erweiterung vollzogen werden.

Mit den Begriffserklärungen Arbeitengraph (Def. A Kap. II.2) und Bearbeitungssystem (Def. C Kap. II.2) sind alle Angaben gemacht, die benötigt werden, um einen Ablaufplan für die Erledigung der gestellten Aufgabe anfertigen zu können. Die Vorgängerrelation ist durch den Arbeitengraph festgelegt, alle anderen Angaben sind im Bearbeitungssystem spezifiziert. Beide Bestandteile bilden gemeinsam das sogenannten Arbeitensystem (Tasksystem).

Def. E (Kap. II.2): Sei $G = (T, <)$ ein Arbeitengraph und $B = (\underline{\tau}, \underline{\mu}, \tau_f, \tau_e, \kappa)$ ein Bearbeitungssystem mit m Prozessoren und s teilbaren Betriebsmitteln. Dann heißt das Tupel (G, B) ein Arbeitensystem (Tasksystem). Es heißt homogen bzw. p-inhomogen, wenn das zugehörige Bearbeitungssystem diese Eigenschaften besitzt.

Der Begriff des Ablaufplans (Schedules) wird im folgenden über mehrere Stufen eingeführt. Dabei erweist es sich im Rahmen dieser Arbeit als sinnvoll, nicht den Begriff des Listenschedules [Coffman & Denning 73] zu verwenden. Es gibt hierfür mehrere Gründe. Zum einen stellt er eine echte Beschränkung bei Arbeitssystemen mit vorgeschriebenen Bearbeitungsintervallen in Verbindung mit Forderungen nach teilbaren Betriebsmitteln dar. Das wird in einem späteren Kapitel gezeigt. Zum anderen ist ein Listenschedule nicht geeignet, eine unterbrechende Bearbeitung zu beschreiben und entzieht sich auch einer knappen formalen Definition, die unumgänglich ist, wenn die mathematische Struktur von Zuordnungsproblemen beleuchtet werden soll. Daß er dennoch eine gewichtige Rolle spielt, hängt mit Implementierungsproblemen zusammen. Er bietet die einfache Möglichkeit, in Mehrprozessoranlagen Strategien dezentral zu verfolgen, die auf statischen Prioritäten [Gonzalez 71] basieren. Hierzu muß die Prioritätsliste lediglich in einem Bereich des Arbeitsspeichers geführt werden, der allen Prozessoren zugänglich ist. Bei einer derartigen Verfahrensweise schränkt man sich jedoch von vorneherein auf im allgemeinen nicht optimale Abläufe ein, wie das spätere Beispiel zeigen wird. Um das in der Praxis zu vermeiden, müssen dynamische Prioritäten in Verbindung mit prozessorspezifischen Schwellwerten eingeführt werden, bei deren Erreichen Prozessoren frühestens Arbeiten aufnehmen dürfen. Dazu an späterer Stelle mehr. Hier soll genügen, daß Listenschedules Zuordnungsprobleme nicht vollständig erfassen und daß daraus auch für die Praxis ein qualitativer Unterschied zwischen Strategien resultiert, die auf statische Prioritäten aufbauen und solchen, die mit dynamischen Prioritäten arbeiten.

Def. F (Kap. II.2): Sei (G,B) ein Arbeitssystem. Eine Abbildung

$$\sigma : T \rightarrow \mathbb{R}^{\geq 0} \times \{1, 2, \dots, m\}$$

$$a \mapsto (\sigma_1(a), \sigma_2(a))$$

heißt nichtunterbrechender Ablaufplan (Schedule) des Arbeitssystems (G,B) .

Die Abbildung $\underline{g}$ beschreibt mit ihrer ersten Komponente σ_1 den Beginnzeitpunkt der Arbeitserledigung und legt in der zweiten Komponente σ_2 den Prozessor fest, auf dem sie durchgeführt wird. Beginnt der Prozessor $\sigma_2(a)$ im Zeitpunkt $\sigma_1(a)$ die Bearbeitung von $a \in T$, dann ist er im halboffenen Intervall $[\sigma_1(a), \sigma_1(a) + \tau_{\sigma_2(a)}(a))$ durch die Arbeit $a \in T$ belegt.

Es ist klar, daß durch diese Festlegung des Begriffs Ablaufplan noch keine ausreichende Charakterisierung des gestellten Problems gefunden ist. Die gegebene Definition reicht zur Beschreibung z.B. deshalb nicht aus, weil in einem Zeitpunkt mehrere Arbeiten einem einzelnen Prozessor zugeteilt sein können und nicht ausgeschlossen ist, daß eine Arbeit a einem Prozessor $\sigma_2(a)$ zugeordnet wird, auf dem sie nicht bearbeitbar ist. Auch die Vorgängerrelation, Forderungen nach teilbaren Betriebsmitteln und dergleichen sind dabei keineswegs berücksichtigt. Die Definition muß weiter verfeinert werden. Das kann in mehreren Richtungen geschehen. Untersuchungen im Bereich des Processor Sharing [Coffmann & Denning 73] betrachten einen Prozessor als teilbares Betriebsmittel und lassen durchaus zu, daß an mehreren Arbeiten gleichzeitig gerechnet wird, dann aber mit verminderter Leistung gegenüber der einzelnen. Wir gehen hier den Weg in Richtung zulässiger Ablaufpläne [Ecker 77], einen Weg, der geeignet ist, die Ergebnisse dieser Arbeit transparent zu machen. Eines läßt sich aus der gewählten Def. F (Kap. II.2) für nichtunterbrechende Ablaufpläne $\underline{g}$ schon jetzt ableiten, nämlich die sogenannten Prozessorfunktionen ξ_i ($1 \leq i \leq m$).

Def. G (Kap. II.2): Sei $\underline{g}$ nichtunterbrechender Ablaufplan des aus m Prozessoren bestehenden Arbeitensystems (G, B) .

Dann heißen die m Abbildungen

$$\xi_i : \mathbb{R}^{\geq 0} \rightarrow P(T) \quad P(T) : \text{Potenzmenge von } T, \quad 1 \leq i \leq m$$

$$t \rightsquigarrow \{ a \in T : \sigma_2(a) = i \wedge \sigma_1(a) \leq t < \sigma_1(a) + \tau_1(a) \}$$

Prozessorfunktionen zum Ablaufplan $\underline{\sigma}$ des Arbeitensystems (G, B) .

Die ξ_i ($1 \leq i \leq m$) beschreiben die zeitliche Belegung des Prozessors i durch Aufgaben. In ihnen kommt die nichtunterbrechende Bearbeitung zum Ausdruck, die beispielhaft dem Bild II.2 entnommen werden kann. Über die Prozessorfunktionen ist es möglich, den bisher definierten Begriff des nichtunterbrechenden Schedules so einzuschränken, wie es die hier verfolgten Problemstellungen erfordern. Der Einfachheit halber wird im folgenden nur noch von Ablaufplänen gesprochen, obwohl nichtunterbrechende gemeint sind.

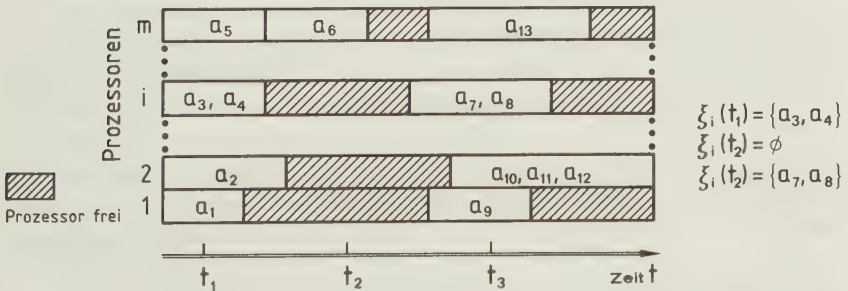


Bild II.2: Gantt-Diagramm eines nichtunterbrechenden Ablaufplans

Def. H (Kap. II.2): Sei $\underline{g}$ Ablaufplan (nichtunterbrechender) eines Arbeitensystems (G, B) mit m Prozessoren und s teilbaren Betriebsmitteln. $\underline{g}$ heißt zulässig, wenn es folgenden Bedingungen genügt:

$$(i) \quad \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq i \leq m \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ t \in \mathbb{R}_{\geq 0} \end{array} \quad |\xi_i(t)| \leq 1$$

$$(ii) \quad \begin{array}{c} \diagup \quad \diagdown \\ a \in T \end{array} \quad \tau_f(a) \leq \sigma_1(a) \wedge \sigma_1(a) + \tau_{\sigma_2(a)}(a) \leq \tau_e(a)$$

$$(iii) \quad \begin{array}{c} \diagup \quad \diagdown \\ a, b \in T \end{array} \quad (a < b \Rightarrow \sigma_1(b) \geq \sigma_1(a) + \tau_{\sigma_2(a)}(a))$$

$$(iv) \quad \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq l \leq s \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ t \in \mathbb{R}_{\geq 0} \end{array} \quad \sum_{i=1}^m \mu_1(\xi_i(t)) \leq 1$$

(Summe über alle i mit $\xi_i(t) \neq \emptyset$)

In der Bedingung (i) ist festgehalten, daß alle Prozessoren zu jedem Zeitpunkt höchstens eine Arbeit erledigen, womit Processor Sharing ausgeschlossen wird. Bedingung (ii) sorgt dafür, daß die Durchführung einer Arbeit frühestens in ihrem Freigabezeitpunkt beginnt und spätestens zum Beendigungszeitpunkt abgeschlossen sein muß. Aufgrund dieser Bedingung können Arbeiten a nur von Prozessoren i ausgeführt werden, für die $\tau_i(a) \neq \infty$ gilt. Ist b Nachfolgearbeit von a , so kann b erst begonnen werden, wenn a bereits erledigt ist. Wegen (iii) erfüllen zulässige Ablaufpläne die im Arbeitsgraph festgelegte Vorgängerbeziehung. In Ergänzung hierzu läßt die Bedingung (iv) die teilbaren Betriebsmittel in jedem Zeitpunkt t maximal zu 100 % auslasten. Mathematisch ist die Schreibweise in (iv) nicht exakt, da μ_1 auf T und nicht auf der Potenzmenge von T definiert ist. Wegen (i) genügt es jedoch, alle einelementigen Teilmengen von T zu betrachten, die kanonisch mit den Elementen von T identifiziert werden können. Ist $\cdot^+$ diese kanonische Abbildung, so müßte (iv) genauer lauten:

$$(iv)^* \quad \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq l \leq s \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ t \in \mathbb{R}_{\geq 0} \end{array} \quad \sum_{i=1}^m \mu_1(\cdot^+(\xi_i(t))) \leq 1$$

Dieser Zusammenhang wird im folgenden nicht mehr explizit erwähnt.

Mit der Definition des zulässigen Ablaufplans ist eine der beiden zentralen Fragestellungen dieser Arbeit formalisiert. Ablaufpläne, die Vorgängerrelationen und sowohl Anzahl als auch Typ der zur Bearbeitung vorhandenen Prozessoren einhalten, sowie den Forderungen nach teilbaren Betriebsmitteln nachkommen und überdies alle Arbeiten ausschließlich innerhalb der vorgegebenen Intervallgrenzen erledigen, existieren offenbar genau dann, wenn sie zulässig sind im Sinn obiger Definition H (Kap. II.2). Solche findet man immer, wenn die Funktionen

$$\tau_f(a) \equiv 0 \text{ und } \tau_e(a) \equiv \sum_{i=1}^n \max\{\tau_j(a_i) : 1 \leq j \leq m\}$$

erfüllen. Unter diesen Voraussetzungen kann ein zulässiger Ablaufplan bestimmt werden, der den Arbeitengraph streng sequentiell berechnet und somit sogar

$$\bigwedge_{t \in \mathbb{R}^{\geq 0}} \sum_{i=1}^m |\xi_i(t)| \leq 1 \quad \text{erfüllt.}$$

Selbstverständlich können τ_f und τ_e noch weiter abgeschwächt werden, ohne ihren trivialen Charakter zu verlieren: indem z.B. die "Maximalabstände" jeder Arbeit a zu Start- und Endarbeiten mit $\tau_f(a)$ und $\tau_e(a)$ übereinstimmen. Das Bearbeitungsintervall $[\tau_f(a), \tau_e(a)]$ ist dann eine Konsequenz des Arbeitengraphs.

Wir wollen uns nun der Formalisierung der zweiten Problemstellung widmen, der Frage nach dem Finden kostenoptimaler Ablaufpläne.

Def. I (Kap. II.2): Sei Σ die Menge aller zulässigen Ablaufpläne eines Arbeitensystems (G, B) mit der Kostenfunktion κ und überdies $\Sigma \neq \emptyset$. Die Abbildung

$$\beta : \Sigma \rightarrow \mathbb{R}$$

$$\underline{\sigma} \mapsto \kappa(\sigma_1(a_i) + \tau_{\sigma_2(a_i)}(a_i)) \quad 1 \leq i \leq n$$

heißt Bewertung der Ablaufpläne von (G, B) bzgl. κ . Ein $\sigma_0 \in \Sigma$ ist optimal, falls

$$\beta(\underline{\sigma}_0) \leq \min\{\beta(\underline{\sigma}) : \underline{\sigma} \in \Sigma\}$$

κ ermittelt Kosten in Abhängigkeit von den Fertigstellungszeitpunkten $\sigma_1(a_i) + \tau_{\sigma_2(a_i)}(a_i)$ jeder Arbeit $a_i \in T$ ($1 \leq i \leq n$)

gemäß einem vorgegebenen Ablaufplan $\underline{\sigma}$. Diese Kosten sind Bewertungskriterium für die Güte eines zulässigen Ablaufplans. Es ist nun einsichtig, daß die Kostenfunktionen $\kappa_2(x_1, x_2, \dots, x_n)$

$$= \frac{1}{n} \sum_{i=1}^n x_i, \quad \kappa_3(x_1, x_2, \dots, x_n) = \frac{1}{n} \sum_{i=1}^n g(a_i) x_i \quad \text{mit } g(a_i) \in \mathbb{R} \quad \text{für}$$

alle $1 \leq i \leq n$ einen Ablaufplan nach seinem mittleren ungewichteten und gewichteten Fluß bewerten [Coffmann & Denning 73]. Die Kostenfunktion $\kappa_1(x_1, x_2, \dots, x_n) = x_{i_0}$ mit $1 \leq i_0 \leq n$

bewertet nach dem Zeitpunkt, zu dem die Arbeit a_{i_0} fertigge-

stellt wird. Sie ist wie κ_2 Spezialfall von κ_3 und insbesondere geeignet, Ablaufpläne gemäß Gesamtverarbeitungszeit des Arbeitensystems zu beurteilen, ein Kriterium, das in der Literatur [Ecker 77] oft auch mit "Länge eines Ablaufplans" bezeichnet wird. a_{i_0} muß zu diesem Zweck die einzige Endarbeit des

Arbeitengraphen sein. Falls mehrere solche Endarbeiten vorhanden sind und trotzdem ein längenminimaler Ablaufplan bestimmt werden soll, bietet sich eine Transformation von (G, B) in ein Arbeitensystem (G', B') an, das nur eine Endarbeit hat.

Hierauf kann dann die Ablaufplanbewertung mittels κ_1 angewandt werden, die einen optimalen Ablaufplan genau dann liefert, wenn er eingeschränkt auf (G,B) ebenfalls optimal ist und somit das ursprünglich gestellte Problem löst.

G' ergibt sich aus G durch Ergänzung um eine Arbeit a_{n+1} , die direkter Nachfolger aller Endarbeiten von G wird. Ihre Rechenzeit wird auf allen Prozessoren mit 1 festgelegt, Betriebsmittelanforderungen, Freigabe- und Endzeitpunkte trivialisiert. Formal sieht das folgendermaßen aus:

$$T' = T \cup \{a_{n+1}\}$$

$$<' = < \cup \{a_i, a_{n+1}\} : a_i \text{ Endarbeit in } G\}$$

$$\underline{r}'(a_{n+1}) = (1, 1, \dots, 1)$$

$$\underline{\mu}'(a_{n+1}) = (0, 0, \dots, 0)$$

$$\tau_f'(a_{n+1}) = 0$$

$$\tau_e'(a_{n+1}) = 1 + \sum_{i=1}^n \max \{\tau_j(a_i) : 1 \leq j \leq m\}$$

$$\kappa' = \kappa_1$$

Für einen zulässigen Ablaufplan $\underline{\sigma}'$ des Arbeitensystems (G', B') stellt $\underline{\sigma} = \underline{\sigma}'|T$ (Restriktion von $\underline{\sigma}'$ auf T) einen zulässigen Ablaufplan von (G, B) dar. Umgekehrt kann zu jedem zulässigen $\underline{\sigma}$ von (G, B) ein zulässiges $\underline{\sigma}'$ von (G', B') bestimmt werden, dessen Restriktion auf T mit $\underline{\sigma}$ übereinstimmt:

$$\underline{\sigma}' : T' \rightarrow \mathbb{R}^{\geq 0} \times \{1, 2, \dots, m\}$$

$$a \rightsquigarrow \begin{cases} \underline{\sigma}(a), & \text{falls } a \in T \\ (\max \{\sigma_1(a) + \tau_{\sigma_2(a)}(a) : a \text{ Endarbeit in } G\}, 1) & \\ \text{sonst.} \end{cases}$$

Sind β_1 bzw. β_1' die Bewertung der Längen derart auseinander abgeleiteter Ablaufpläne, so gilt:

$$\beta_1(\underline{\sigma}) = \beta_1'(\underline{\sigma}') - 1$$

Das heißt, der längenoptimale Ablaufplan von (G, B) kann aus dem von (G', B') direkt bestimmt werden, letzterer ist aber über die Kostenfunktion κ_1 ermittelbar.

Obwohl κ_1 und κ_2 Spezialfälle von κ_3 , der gewichteten mittleren Flußzeit sind, konnten viele Ergebnisse nur für κ_1 bzw. κ_2 gezeigt werden. Bezüglich κ_3 optimale Ablaufpläne konnten bisher nur für 1-Prozessorsysteme [Horn 72] oder für homogene m-Prozessorsysteme bei unabhängigen Arbeiten [Rothkopf 66], [Lawler & Moore 69] bestimmt werden. Weitere Kostenfunktionen sind

$$\kappa_4(x_1, x_2, \dots, x_n) = \sum_{i=1}^n (x_i - d(a_i)), \text{ wobei } d(a_i) \in \mathbb{R}^{\geq 0} \quad (1 \leq i \leq n)$$

$$\kappa_5(x_1, x_2, \dots, x_n) = \sum_{i=1}^n \max \{0, x_i - d(a_i)\} \quad - " -$$

$$\kappa_6(x_1, x_2, \dots, x_n) = \sum_{i=1}^n u_i, \text{ wobei } u_i = 0, \text{ falls } x_i \leq d(a_i) \text{ und} \\ u_i = 1, \text{ falls } x_i > d(a_i)$$

$$\kappa_7(x_1, x_2, \dots, x_n) = \max \{x_i - d(a_i) : 1 \leq i \leq n\}$$

$$\kappa_8(x_1, x_2, \dots, x_n) = \max \{\max (0, x_i - d(a_i)) : 1 \leq i \leq n\}$$

$$\kappa_9(x_1, x_2, \dots, x_n) = \max \{u_i : 1 \leq i \leq n\} \text{ mit}$$

$$u_i = \begin{cases} 0, & \text{falls } x_i \leq d(a_i) \\ 1 & \text{sonst.} \end{cases}$$

Sie beschreiben die Verspätung und das Überschreiten von Soll-Fertigstellungszeitpunkten $d(a_i)$ der auszuführenden Aufgaben a_i ($1 \leq i \leq n$). Im Verlauf dieser Arbeit werden Ergebnisse erzielt, die sich auf alle genannten Kostenfunktionen anwenden lassen, weil diese die Isotonie-Eigenschaft besitzen.

Mit den eingeführten formalen Festlegungen sind beide Fragestellungen präzisiert. Die Existenz eines zulässigen Ablaufplanes für ein gegebenes Arbeitensystem wird im weiteren durch

$$(G, B, \text{ex. } \underline{g}?)$$

abgekürzt, das Problem, im Existenzfall einen κ -optimalen Ablaufplan zu bekommen, mit

$$(G, B, \text{opt}_{\kappa}).$$

Diese Kurzschreibweisen lehnen sich an [Ecker 77] an. Bevor beide Probleme mit analytischen und algebraischen Methoden untersucht werden, ein kurzer Überblick der bisher erzielten Ergebnisse, die zum großen Teil [Ecker 77] und [Graham et al. 78] entnommen und aktualisiert wurden.

II.3 Ergebnisse zu den Problemen $(G, B, \text{ex. } \underline{g}?)$ und $(G, B, \text{opt}_{\kappa})$

Die fünf Übersichten in den Bildern II.3 - 7 geben den momentanen Stand der Ergebnisse zu obigen beiden Fragestellungen wieder. Die Existenzfrage für zulässige Ablaufpläne (Bild II.3) ist selbst bei 1-Prozessorsystemen nicht vollständig beantwortet.

ARBEITENSYSTEM						LITERATUR		BEMERKUNGEN	
ARBEITEN- GRAPH	PROZESSOREN- SYSTEM	UNTERR- BRECHUNG	ANZAHL DER PROZESSOREN	RECHEN- ZEIT	FREIGABE- ZEIT	BEENDIGUNGS- ZEIT			
bel.	hom.		1	bel.	0	bel.	[Lawler 73]	$O(n^2)$	*
bel.	hom.		1	1	bel.	bel.	[Lageweg et al. 76]	$O(p(x))$	*
bel.	hom.	ja	1	bel.	bel.	bel.	[Lageweg et al. 76]	$O(p(x))$	*
$\emptyset$	hom.		1	bel.	0	bel.	[Lawler 75]	$O(n^2 \sum \tau_i)$	*
$\emptyset$	hom.		1	1	bel.	bel.	[Lawler 75]	$O(p(x))$	*
$\emptyset$	hom.		1	bel.	0	bel.	[Moore 68] [Sidney 73]	$O(n \log n)$	*
bel.	hom.		1	1	bel.	bel.	[Baker & Su 74] [Mahon & Florian 75] [Lageweg et al. 76]	abzählender Algorithmus, brauchbar bei bis zu 80 Arbeiten	*
$\emptyset$	hom.		1	bel.	0	bel.	[Kincooy et al. 75] [Gelders & Kleindorfer 74/75] [Fisher 76] [Baker & Schrage 76]	abzählender Algorithmus, brauchbar bei bis zu 50 Arbeiten	*
bel.	hom.		2	1	$\in N$	$\in N$	[Carey & Johnson 76]	$O(p(x))$	
Baum	hom.		bel.	1	0	$\in N$	[Brucker et al. 77]	$O(p(x))$	
$\emptyset$	unif.		bel.	bel.	0	bel.	[Rothkopf 66] [Lawler & Moore 69]	abzählender Algorithmus	*

Bild II.3 Ergebnisse zu (G,B,ex.07)

*) Diese Ergebnisse sind Nebenprodukte von Algorithmen, die optimale Ablaufpläne bzgl. den Kostenfunktionen $K = \max(x_i - d(a_i))$ bzw. $K = \max(\max(0, x_i - d(a_i)))$ erstellen.

ARBEITENSYSYSTEM						
KOSTEN- FUNKTION	ARBEITEN- GRAPH	UNTER- BRECHUNG	ANZAHL D. PROZESSOREN	RECHENZEIT	FREIGABE- ZEIT	BEENDIGUNGS- ZEIT
$\kappa = \max \kappa_i(x_i)$ κ_i isoton	bel.		1	bel. triv.	triv.	[Lawler 73]
$\kappa = \max \kappa_i(x_i)$ $\kappa_i(x_i) = x_{i-1} - d(a_i)$	bel.		1	1	bel. triv.	$0(n^2)_i, \kappa_i = x_{i-1} - d(a_i)$ Jackson's Regel [Jackson 55]
$\kappa = \max \kappa_i(x_i)$ $\kappa_i(x_i) = x_{i-1} - d(a_i)$	bel.	*	1	bel. bel.	triv.	$0(p(x))$
$\kappa = \sum g(a_i) x_i$	$\emptyset$		1	bel. triv.	triv.	$0(p(x))$
$\kappa = \sum g(a_i) x_i$	Baum oder seriell/parallel		1	bel. triv.	triv.	$0(n \log n)$, Smith's Regel $g(a_i) = 1$ = SPT-Algorithmus
$\kappa = \sum x_i$	$\emptyset$	*	1	bel. triv.	triv.	$0(p(x))$
$\kappa = \sum \max\{0, x_{i-1} - d(a_i)\}$	$\emptyset$		1	bel. triv.	triv.	$0(n^4 \sum \tau_i)$
$\kappa = \sum \max\{0, x_{i-1} - d(a_i)\}$	$\emptyset$		1	1	bel. triv.	$0(p(x))$
$\kappa = \sum u_i(x_i)$	$\emptyset$		1	bel. triv.	triv.	$0(n \log n)$
$u_i(x_i) = \begin{cases} 1, & \text{falls } x_i > d(a_i) \\ 0, & \text{sonst.} \end{cases}$						

Bild II.4: Polynomial beschränkte Algorithmen zu $(G, B, \text{opt}_\kappa)$ im 1-Prozessorsystem

ARBEITENSYSYSTEM									
KOSTEN- FUNKTION	ARBEITEN- GRAPH	PROZESSOR- SYSTEM	UNTER- BRECHUNG	ANZAHL DER PROZESSOREN	RECHENZEIT	BETRIEBS- MITTEL	FREIGABE- ZEIT	BEENDIGUNGS- ZEIT	
$\kappa = \max x_i$	bel.	hom.		2	1	triv.	triv.	triv.	L I T E R A T U R
$\kappa = \max x_i$	baumähnlich	hom.		bel.	1	triv.	triv.	triv.	[Coffman & Graham 72] [Fuji & Kasami & Ninomiya 69/71]
$\kappa = \max x_i$	Wald aus Bäumen und A-Bäumen	hom.		3	1	triv.	triv.	triv.	[Hu 61] [Hsu 66]
$\kappa = \max x_i$	$\emptyset$	hom.		2	1	bel.	triv.	triv.	[Ladtko 77]
$\kappa = \max x_i$	flow-shop	inhom.		2 (3)	bel.	triv.	triv.	triv.	[Carey & Johnson 75]
$\kappa = \sum x_i$	$\emptyset$	hetero- gen		bel.	bel.	triv.	triv.	triv.	[Johnson 54]
									[Conway & Maxwell & Miller 67] [Bruno & Coffman & Sethi 74] [Horn 73] [Clark 74]

Bild II.5: Polynomial beschränkte Algorithmen zu (G, B, opt_K)

für m-Processorsysteme ($m > 1$)

A R B E I T E N S Y S T E M										
KOSTENFUNKTION	ARBEITENSGRAPH	PROZESSOREN-SYSTEM	UNTERBRECHUNG	ANZAHL DER PROZESSOREN	RECHENZEIT	BETRIEBS-MITTEL	FREIGABE-ZEIT	BEENDIGUNGS-ZEIT	L I T E R A T U R	BEMERKUNGEN
$\kappa = \max(x_i - d(a_i))$	bel.	hom.		1	1	-	bel.	triv.	[Baker & Su 74][Lageweg et al. 76] [Mahon & Florian 75]	brauchbar bei bis zu 80 Arb.
$\kappa = \sum g(a_i) \max\{0, \kappa_i(x_i)\}$ $\kappa_i(x_i) = x_i - d(a_i)$	$\emptyset$	hom.		1	bel.	-	triv.	triv.	[Rinnooy et al. 75][Fisher 76] [Gelders & Kleindorfer 74/75] [Baker & Schrage 76]	brauchbar bei bis zu 50 Arb.
$\kappa = \max x_i$	$\emptyset$	hom.		bel.	bel.	bel.	bel.	bel.	[Stern 76] [Bradley et al. 75]	branch-and-bound-Verf.
$\kappa = \max x_i$	$\emptyset$	hom.		bel.	bel.	triv.	bel.	triv.		
$\kappa = \max(x_i - d(a_i))$	$\emptyset$	unif.		bel.	bel.	triv.	triv.	triv.		
$\kappa = \sum g(a_i) x_i$	$\emptyset$	unif.		bel.	bel.	triv.	triv.	triv.		$0(m \cdot n \cdot c^{w-1})$ falls alle $d(a_i)$ gleich $\approx$ $0(m \cdot \min\{3^n, 2^n \cdot c\})$
$\kappa = \sum g(a_i) n_i$ $u_i = \begin{cases} 1, & \text{falls } x_i > d(a_i) \\ 0 & \text{sonst.} \end{cases}$	$\emptyset$	hom.		bel.	bel.	triv.	triv.	triv.	[Rothkopf 66][Lawler & Moore 69]	
$\kappa = \sum g(a_i) \max\{0, x_i(x_i)\}$ $\kappa_i = x_i - d(a_i)$	$\emptyset$	hom.		bel.	bel.	triv.	triv.	triv.		
$\kappa = \sum \kappa_i(x_i)$	$\emptyset$	hom.		bel.	bel.	triv.	triv.	triv.		
$\kappa = \max \kappa_i(x_i)$ κ_i isoton	$\emptyset$	hom.		bel.	bel.	triv.	triv.	triv.		

Bild II.6: Abzählende Algorithmen für $(G, B, \text{opt}_\kappa)$

ARBEITENS SYSTEM						BEMERKUNG
KOSTEN- FUNKTION	ARBEITEN- GRAPH	UNTER- BRUCHUNG	ANZAHL DER PROZESSOREN	RECHENZEIT	FRIGABE- ZEIT	
$\kappa = \max(x_i - d(a_i))$	bel.		1	bel.	triv.	[Lengstra et al. 75]
$\kappa = \sum [g(a_i)] x_i$	bel.		1	1 [bel.]	triv.	[Lawler 76] [Lengstra & Rinnooy 76]
$\kappa = \sum x_i$	$\emptyset$		1	bel.	triv.	[Lengstra et al. 75]
$\kappa = \sum g(a_i) x_i$	$\emptyset$	*	1	bel.	triv.	[Lengstra -]
$\kappa = \sum g(a_i) \max\{0, x_i(x_i)\}$ $\kappa_i(x_i) = x_i - d(a_i)$	$\emptyset$		1	bel.	triv.	[Lawler 75] [Lengstra et al. 75]
$\kappa = \sum \max\{0, x_i(x_i)\}$ $\kappa_i(x_i) = x_i - d(a_i)$	bel.		1	1	triv.	[Lengstra & Rinnooy 76]
$\kappa = \sum \max\{0, x_i(x_i)\}$ $\kappa_i(x_i) = x_i - d(a_i)$	$\emptyset$		1	bel.	triv.	[Lengstra & Rinnooy 76]
$\kappa = \sum g(a_i) u_i$ $u_i = \begin{cases} 1, & \text{falls } x_i \geq d(a_i) \\ 0, & \text{sonst.} \end{cases}$	$\emptyset$		1	bel.	triv.	[Karp 72] [Lawler & Moore 69]
$\kappa = \sum n_i$ $u_i = \begin{cases} 1, & \text{falls } x_i > d(a_i) \\ 0, & \text{sonst.} \end{cases}$	bel.		1	1	triv.	[Garey & Johnson 76]

Bild II.7: NP-vollständige Probleme im 1-Prozessorsystem

Entweder muß der Freigabezeitpunkt jeder Arbeit im Ursprung liegen oder die Bearbeitungszeit stimmt mit dem Einheitsintervall überein. Bei Mehrprozessorsystemen sind zusätzliche Einschränkungen entweder über die Anzahl der Prozessoren (Beschränkung auf 2) oder über den Arbeitengraph (Baum) nötig. Selbst abzählende Algorithmen erfassen heterogene Prozessorsysteme nicht, sondern bleiben auf uniforme beschränkt, und das auch nur bei unabhängigen Arbeiten. Eine übergreifende Charakterisierung der Existenzfrage von zulässigen Ablaufplänen existiert derzeit nicht.

Alle polynomial beschränkten Algorithmen zur Bestimmung kostenoptimaler Abläufe sind in den Bildern II.4 (1-Prozessorsysteme) und II.5 (m-Prozessorsysteme mit $m \geq 2$) aufgeführt. Schon das Aufsuchen von Monoprozessor-Ablaufplänen muß schwierig sein, da deren Finden selbst bei unabhängigen Arbeiten in zahlreichen Fällen ein NP-vollständiges Problem ist (z.B. [Lengstra et al. 75] oder [Lengstra & Rinnooy 76]). Bild II.7 beschreibt diese Fälle derzeit vollständig. Insofern überrascht es nicht, daß polynomial beschränkte Algorithmen für heterogene Prozessorsysteme nur unter ganz bestimmten Einschränkungen bislang bestimmt werden konnten. Betriebsmittelanforderungen, Freigabe- und Beendigungszeitpunkte mußten trivialisiert werden und außerdem die Arbeiten unabhängig voneinander sein (z.B. [Bruno et al. 74]). Erstaunlich ist demgegenüber die Schärfe der Voraussetzungen, die dann noch gemacht werden müssen, wenn abzählende Algorithmen geeignet sein sollen, kostenminimale Ablaufpläne zu bestimmen. Mit einer einzigen Ausnahme ([Baker & Su 74], [Lageweg 76], [Mahon & Florian 75]) müssen die zu erledigenden Arbeiten unabhängig voneinander sein (Bild II.6) und dürfen weder Betriebsmittelforderung stellen, noch innerhalb vorgegebener Zeitschranken ablaufen. Heterogene Prozessorsysteme entziehen sich solchen Verfahren, ja sogar inhomogene. Nur zwei Ergebnisse existieren für den Fall, daß die Prozessoren feste Leistungsfaktoren zueinander einhalten, uniform sind ([Rothkopf 66], [Lawler & Moore 69]).

Über die beiden hier erörterten Fragestellungen hinaus sind Resultate bei inhomogenen Prozessorsystemen nicht bekannt, mit zwei Ausnahmen, die das "worst case" Verhalten betreffen. Das schließt auch heuristische Algorithmen mit einem Gütemaß ein, von denen somit bekannt ist, wieweit sie maximal von einem optimalen Ablaufplan abweichen. Die beiden Ausnahmen beziehen sich auf inhomogene Prozessorsysteme mit zwei Typen. Es ist trivial, daß unter der Voraussetzung, von jedem Typ genau ein Exemplar zu haben, die maximale Zeitdifferenz zweier Ablaufpläne durch den Faktor 2 begrenzt ist. Bestenfalls arbeiten sie gleichzeitig, schlechtestenfalls unter gegenseitigem Ausschluß. Daß die Schranke 2 die bestmögliche ist, zeigt das einfache Beispiel von Bild II.8.

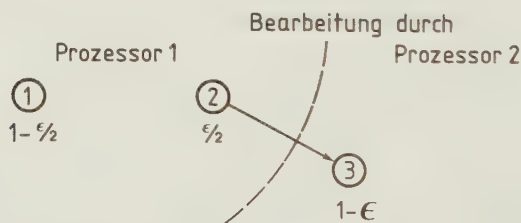


Bild II.8: Bestmögliche Schranke 2

Eine ähnliche Abschätzung existiert nur noch für den Fall, daß wiederum 2 Prozessortypen vorhanden sind, diesmal aber von einem Typ zwei Exemplare, vom anderen Typ eins. Der Faktor beträgt dann $5/2$ (mündliche Überlieferung von C.L. Liu und J.W.S. Liu gemäß [Ecker 77]). Hieran wird klar, daß inhomogene und somit erst recht heterogene Prozessorsysteme im Vergleich zu homogenen (Faktor 2 [Graham 66], [Graham 69]) mehr Spielraum an Rechenzeit für das Bestimmen optimaler Ablaufpläne zur Verfügung stellen.

KAPITEL III: LÖSBARKEIT VON $(G, B, \text{ex. } \underline{\sigma}?)$ UND (G, B, opt_μ)

MIT PSEUDO-BOOLEE METHODEN

III.1 Dualität Prozessorfunktion-Arbeitsfunktion

Im Kapitel II wurde definiert, was unter einem zulässigen Ablaufplan verstanden wird. Das geschah mittels der Prozessorfunktionen ξ_i ($1 \leq i \leq m$: Anzahl der Prozessoren), die prozessor-spezifisch und zeitabhängig festhalten, welche Arbeiten gerade erledigt werden. Sie sind eine Formalisierung des intuitiven Verständnisses über den zeitlichen Ablauf bei der Durchführung eines Arbeitengraphen, und zwar aus der Sicht der Prozessoren. Versucht man damit eine analytische Charakterisierung der Zuordnungsprobleme $(G, B, \text{ex. } \underline{\sigma}?)$ und (G, B, opt_μ) , so ist bereits aus Def. H (Kap. II.2) erkennbar, daß dies mit ihnen allein nicht geht. Die Vorgängerrelation im Arbeitengraphen und die über die Knotenbewertungen τ_f und τ_e eingebrachten Bearbeitungsintervalle sind einzig über die Komponenten σ_1 und σ_2 eines Ablaufplans erfaßt ((ii) und (iii)) von Def. H (Kap. II.2)), dessen Festlegung bewußt allgemein gehalten wurde. Aus dieser Schwierigkeit kommt man heraus, wenn das Vorgehen gemäß eines Ablaufplans nicht aus der Sicht der Prozessoren, sondern von den Aufgaben her betrachtet wird. Zu diesem Zweck führen wir den Begriff der Arbeitsfunktion ein und geben im folgenden eine äquivalente Darstellung für die Zulässigkeit eines Ablaufs: in der Terminologie Dijkstra's [Dijkstra 76] ein nicht determinierter Prädikamentransformer für die Zuordnungsprobleme $(G, B, \text{ex. } \underline{\sigma}?)$ und (G, B, opt_μ) .

Def. K (Kap. III.1): Sei $\underline{\sigma}$ Ablaufplan (nicht notwendig zulässig) eines Arbeitensystems (G, B) mit n Arbeiten und m Prozessoren. Für jede Arbeit $a_i \in T$ ($1 \leq i \leq n$) heißen die folgenden Abbildungen Arbeitsfunktionen:

$$\eta_i : \mathbb{R}^{\geq 0} \rightarrow \{0,1\}$$

$$t \mapsto \begin{cases} 1, & \text{falls } \sigma_1(a_i) \leq t < \sigma_1(a_i) + \tau_{\sigma_2(a_i)}(a_i) \\ 0 & \text{sonst.} \end{cases}$$

Die Arbeitsfunktionen sind Intervallindikatorfunktionen, die zeitabhängig den Bearbeitungsstand einer Aufgabe $a_i \in T$ beschreiben. Der Funktionswert 1 von η_i zur Zeit t läßt die Arbeit a_i als "in Bearbeitung" erkennen. Von welchem Prozessor das getan wird, kann aus der Kenntnis der η_i allein nicht mehr rekonstruiert werden, auch dann nicht, wenn die Rechenzeiten aller Arbeiten bekannt sind. Hierzu würde $\underline{\sigma}$ benötigt. Es zeigt sich jedoch, daß diese Kenntnis zur Charakterisierung von Zuordnungsproblemen nicht notwendig ist.

Setzen wir voraus, daß die η_i ($1 \leq i \leq n$) Arbeitsfunktionen eines zulässigen Ablaufplans sind, so verfügen sie über bestimmte Eigenschaften.

Eigenschaft 1:

$$\bigwedge_{t \in \mathbb{R}^{\geq 0}} \sum_{i=1}^n \eta_i(t) \leq m \quad (m: \text{Anzahl der Prozessoren})$$

Aufgrund der Bedingung (i) von Def. H (Kap. II.2) erhält man

$$(*) \quad \bigwedge_{t \in \mathbb{R}^{\geq 0}} \sum_{i=1}^m |\xi_i(t)| \leq m$$

Zusätzlich weiß man

$$\bigwedge_{t \in \mathbb{R}^{\geq 0}} \bigwedge_{1 \leq i \leq m} (|\xi_i(t)| = 1 \Rightarrow \bigvee_{1 \leq j \leq n} \eta_j(t) = 1 \wedge \sigma_2(a_j) = i)$$

und umgekehrt

$$\bigwedge_{t \in \mathbb{R}} t \geq 0 \quad \bigwedge_{1 \leq j \leq n} (\eta_j(t) = 1 \Rightarrow \bigvee_{1 \leq i \leq m} \xi_i(t) = \{a_j\})$$

$$\text{d.h.} \quad \bigwedge_{t \in \mathbb{R}} t \geq 0 \quad \sum_{i=1}^m |\xi_i(t)| = \sum_{j=1}^n \eta_j(t)$$

Zusammen mit (*) ergibt sich Eigenschaft 1.

Eigenschaft 2:

$$\bigwedge_{1 \leq i \leq n} \int_0^{\infty} \eta_i(t) dt = \tau_{\sigma_2(a_i)}(a_i) = \frac{\tau_e(a_i)}{\tau_f(a_i)} \int_0^{\infty} \eta_i(t) dt$$

Die erste Gleichheit folgt unmittelbar aus der Definition der η_i (Def. K (Kap. III.1)), die zweite gilt bei zulässigen Ablaufplänen wegen (ii) von Def. H (Kap. II.2).

Eigenschaft 3:

$$\begin{aligned} \bigwedge_{a_i, a_j \in T} (a_i < a_j \Rightarrow \int_0^{\infty} t (\tau_{\sigma_2(a_i)}(a_i) \eta_j(t) - \tau_{\sigma_2(a_j)}(a_j) \eta_i(t)) dt \geq \\ \geq \frac{1}{2} \tau_{\sigma_2(a_i)}(a_i) \tau_{\sigma_2(a_j)}(a_j) (\tau_{\sigma_2(a_i)}(a_i) + \tau_{\sigma_2(a_j)}(a_j))) \end{aligned}$$

Betrachte hierzu für ein $1 \leq i \leq n$ das Integral der Funktion $t\eta_i(t)$ im Intervall 0 bis ∞ . Geometrische Gegebenheiten lassen erkennen (Bild III.1), daß die Gleichheit

$$\int_0^{\infty} t \eta_i(t) dt = \sigma_1(a_i) \tau_{\sigma_2(a_i)}(a_i) + \frac{1}{2} \tau_{\sigma_2(a_i)}(a_i)^2$$

gilt, woraus die Eigenschaft 3 folgt, denn:

$$\begin{aligned} & \int_0^{\infty} t (\tau_{\sigma_2(a_i)}(a_i) \eta_j(t) - \tau_{\sigma_2(a_j)}(a_j) \eta_i(t)) dt = \\ &= \sigma_1(a_j) \tau_{\sigma_2(a_i)}(a_i) \tau_{\sigma_2(a_j)}(a_j) + \frac{1}{2} \tau_{\sigma_2(a_j)}(a_j)^2 \tau_{\sigma_2(a_i)}(a_i) - \\ &- \sigma_1(a_i) \tau_{\sigma_2(a_i)}(a_i) \tau_{\sigma_2(a_j)}(a_j) - \frac{1}{2} \tau_{\sigma_2(a_i)}(a_i)^2 \tau_{\sigma_2(a_j)}(a_j) = \\ &= \tau_{\sigma_2(a_i)}(a_i) \tau_{\sigma_2(a_j)}(a_j) [\sigma_1(a_j) + \frac{1}{2} \tau_{\sigma_2(a_j)}(a_j) - \sigma_1(a_i) - \frac{1}{2} \tau_{\sigma_2(a_i)}(a_i)] \\ &= \tau_{\sigma_2(a_i)}(a_i) \tau_{\sigma_2(a_j)}(a_j) [\underbrace{\sigma_1(a_j) - (\sigma_1(a_i) + \tau_{\sigma_2(a_i)}(a_i))}_{(*)} + \frac{1}{2} \tau_{\sigma_2(a_j)}(a_j) + \\ &+ \frac{1}{2} \tau_{\sigma_2(a_i)}(a_i)] \\ &\geq \tau_{\sigma_2(a_i)}(a_i) \tau_{\sigma_2(a_j)}(a_j) [\frac{1}{2} \tau_{\sigma_2(a_i)}(a_i) + \frac{1}{2} \tau_{\sigma_2(a_j)}(a_j)] \end{aligned}$$

Das Ungleichheitszeichen ist richtig, weil für zulässige Ablaufpläne die Klammer (*) nie kleiner als 0 werden kann. Das garantiert die Bedingung (iii) von Def. H (Kap. II.2).

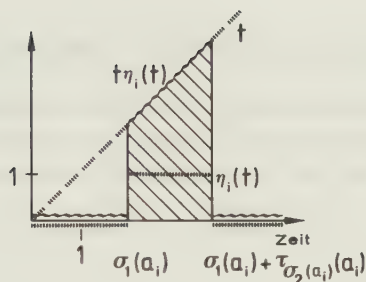


Bild III.1: $\int_0^{\infty} t \eta_i(t) dt$

Eigenschaft 4:

$$1 \leq l \leq s \quad t \in \mathbb{R}^{\geq 0} \quad \sum_{i=1}^n \mu_1(a_i) \eta_i(t) \leq 1$$

wie bei Eigenschaft 1 kann gezeigt werden

$$1 \leq l \leq s \quad t \in \mathbb{R}^{\geq 0} \quad \sum_{i=1}^n \mu_1(a_i) \eta_i(t) = \sum_{i=1}^m \mu_1(\xi_i(t))$$

(2. Summe über alle i mit $\xi_i(t) \neq \emptyset$)

Die obige Ungleichung folgt somit aus der Bedingung (i) und (iv) von Def. H (Kap. II.2).

Im Falle eines homogenen Prozessorsystems verschwinden die Indizes $\sigma_2(a_i)$ und $\sigma_2(a_j)$ aus den Eigenschaften 2 und 3 und man erhält vier notwendige Bedingungen für die Existenz eines zulässigen Ablaufplans, die allein über die Intervallindikatorfunktionen η_i ($1 \leq i \leq n$) ausgedrückt sind. Weder die Komponente σ_1 noch σ_2 spielen dabei eine Rolle. Diese vier Eigenschaften sind aber auch hinreichend für die Lösbarkeit des Zuordnungsproblems (G,B,ex. 9) im Fall eines homogenen und, wie wir später sehen werden, auch p-inhomogenen Prozessorsystems.

Satz 1 (Kap. III.1): Dualität Prozessorfunktion-Arbeitsfunktion

Sei (G, B) ein Arbeitensystem mit homogenem Prozessorsystem. Dann existiert ein zulässiger Ablaufplan $\underline{\sigma}$ genau dann, wenn es n (Anzahl der Arbeiten in G) Intervallindikatorfunktionen

$$\eta_i : \mathbb{R}^{\geq 0} \rightarrow \{0, 1\}$$

gibt, die folgenden Bedingungen genügen:

$$(i) \quad \bigwedge_{t \in \mathbb{R}^{\geq 0}} \sum_{i=1}^n \eta_i(t) \leq m \text{ (Anzahl der Proz.)}$$

$$(ii) \quad \bigwedge_{1 \leq i \leq n} \int_0^{\infty} \eta_i(t) dt = \tau(a_i) = \int_{\tau_F(a_i)}^{\tau_E(a_i)} \eta_i(t) dt$$

$$(iii) \quad \bigwedge_{a_i, a_j \in T} (a_i < a_j \Rightarrow \int_0^{\infty} t(\tau(a_i)\eta_j(t) - \tau(a_j)\eta_i(t)) dt \geq \frac{1}{2} \tau(a_i)\tau(a_j)(\tau(a_i) + \tau(a_j)))$$

$$(iv) \quad \bigwedge_{1 \leq l \leq s} \bigwedge_{t \in \mathbb{R}^{\geq 0}} \sum_{i=1}^n \mu_l(a_i) \eta_i(t) \leq 1 \text{ (s : Anzahl der Betriebsmittel)}$$

Bew.:

1. Schritt: Die n Arbeitsfunktionen η_i ($1 \leq i \leq n$) eines zulässigen Ablaufplans $\underline{\sigma}$ sind Intervallindikatorfunktionen und erfüllen (i) - (iv).

Das ist eine Konsequenz aus den Eigenschaften 1 - 4. Sie sind notwendige Bedingungen für die Existenz eines zulässigen Ablaufplans und stimmen bei homogenen Prozessorfunktionen mit den Bedingungen (i) - (iv) des Satzes überein.

2. Schritt: Aus n Intervallindikatorfunktionen η_i ($1 \leq i \leq n$)

mit den Eigenschaften (i) - (iv) des Satzes kann ein zulässiger Ablaufplan abgeleitet werden, dessen Arbeitsfunktionen mit den η_i identisch sind.

O.B.d.A. sollen die Intervallindikatorfunktionen η_i ($1 \leq i \leq n$) auf einem halboffenen Intervall $[x_i, y_i)$ ($1 \leq i \leq n$) den Wert 1 annehmen. Falls das nicht von vorneherein zutrifft, tun dies die Abbildungen

$$\eta_i^* : \mathbb{R}^{\geq 0} \rightarrow \{0, 1\}$$

$$t \rightsquigarrow \begin{cases} 1, & \text{falls } \inf \{t' \in \mathbb{R}^{\geq 0} : \eta_i(t') = 1\} \leq t < \\ & \sup \{t' \in \mathbb{R}^{\geq 0} : \eta_i(t') = 1\} \\ 0 & \text{sonst} \end{cases}$$

die ebenfalls die Eigenschaften (i) - (iv) erfüllen. (ii) und (iii) gelten wegen der beiden Gleichheiten

$$\int_0^{\infty} \eta_i(t) dt = \int_0^{\infty} \eta_i^*(t) dt$$

$$\int_0^{\infty} t \eta_i(t) dt = \int_0^{\infty} t \eta_i^*(t) dt$$

Um (i) und (iv) nachzuweisen, betrachten wir die endliche Menge X aller $x_i = \inf \{t \in \mathbb{R}^{\geq 0} : \eta_i(t) = 1\}$ und wählen $\Delta x = \min \{|x_i - x_j| : 1 \leq i \neq j \leq n\}$.

Analog sei Y die Menge aller oberen Intervallgrenzen der η_i , also $Y = \{y_i = \sup \{t \in \mathbb{R}^{\geq 0} : \eta_i(t) = 1\} : 1 \leq i \leq n\}$. ΔY sei deren minimaler Abstand: $\Delta Y = \min\{|y_i - y_j| : 1 \leq i \neq j \leq n\}$. Dann gilt für alle $t \in \mathbb{R}^{\geq 0} \setminus (X \cup Y)$:

$$(0) \quad \eta_i^*(t) = \eta_i(t)$$

Sei nun $t \in X$ und $\sum_{i=1}^n \eta_i^*(t) > m$. Dann existiert für alle $1 \leq i \leq n$ mit $\eta_i^*(t) = 1$ (Indexmenge I) ein $\varepsilon_i \in \mathbb{R}^{>0}$ mit den Eigenschaften

$$(1) \quad \eta_i^*(t + \varepsilon_i) = 1$$

$$(2) \quad \varepsilon_i < \min\{\Delta X, \Delta Y\}$$

mit $\varepsilon = \min\{\varepsilon_i : i \in I\}$ folgt aus (1)

$$\bigwedge_{i \in I} \eta_i^*(t + \varepsilon) = 1$$

Wegen (2) gilt: $t + \varepsilon \notin X \cup Y$

$$\text{Also zusammen mit (0): } \bigwedge_{i \in I} \eta_i^*(t + \varepsilon) = \eta_i(t + \varepsilon)$$

Das führt zu einem Widerspruch, denn

$$\sum_{i=1}^n \eta_i(t + \varepsilon) \geq \sum_{i=1}^n \eta_i^*(t + \varepsilon) = \sum_{i=1}^n \eta_i^*(t) > m$$

$$\Rightarrow \bigwedge_{t \in X} \sum_{i=1}^n \eta_i^*(t) \leq m$$

Nun muß noch der Fall $t \in Y \setminus X$ betrachtet werden. Aufgrund der Definition der η_i^* gilt für solche Punkte $\eta_i^*(t) \leq \eta_i(t)$ und damit auch

$$\sum_{i=1}^n \eta_i^*(t) \leq \sum_{i=1}^n \eta_i(t) \leq m$$

Wir konnten also die Intervallindikatorfunktionen o.B.d.A. auf einem halboffenen Intervall den Wert 1 annehmen lassen. Die darauf aufbauende Definition eines zulässigen Ablaufplans geschieht über zwei Mengen, die ausnutzen, daß X in einer aufsteigenden Folge $x_{i_1} < x_{i_2} < \dots < x_{i_k}$ ($k \leq n$) angeordnet werden kann. Für $1 \leq l \leq k$ sei

$$T_1' := \{a_j \in T : \eta_j(x_{i_1}) = 1 \wedge \bigwedge_{\varepsilon > 0} \eta_j(x_{i_1} - \varepsilon) = 0\}$$

$$T_1'' := \{a_j \in T : \eta_j(x_{i_1}) = 1 \wedge \bigvee_{\varepsilon > 0} \eta_j(x_{i_1} - \varepsilon) = 1\}$$

Für den nachfolgend definierten Ablauf beinhaltet T_1' alle Aufgaben, deren Bearbeitung im Zeitpunkt x_{i_1} begonnen wird, T_1'' die Arbeiten, die vor x_{i_1} begonnen wurden und zu diesem Zeitpunkt noch nicht fertig sind. Wegen Voraussetzung (i) gilt:

$$(*) \quad \bigwedge_{1 \leq l \leq k} |T_1'| + |T_1''| \leq m$$

Überdies bildet $\bigcup_{1 \leq l \leq k} T_1'$ eine Partition von T , so daß

rekursiv über l ein Ablaufplan $\underline{g}$ definiert werden kann, dessen erste Komponente durch

$$\bigwedge_{1 \leq i \leq n} \sigma_1(a_i) = x_i$$

bestimmt ist. Die 2. Komponente ermittelt folgender Algorithmus

- ① $l=1; M_f = \{1, 2, \dots, m\}; M_b = \emptyset$
- ② $a \in T_1' ? \Rightarrow (\sigma_2(a) \in M_f \setminus M_b; M_b = M_b \cup \{\sigma_2(a)\})$
- ③ $T_1' \neq \emptyset \Rightarrow$ ②
- ④ $l = l + 1$
- ⑤ $l = k + 1 ? \Rightarrow$ ⑧
- ⑥ $M_b = \{\sigma_2(a) : a \in T_1''\}$
- ⑦ ②
- ⑧ end

Die Eigenschaft (*) gewährleistet, daß im Schritt ② jederzeit ein $\sigma_2(a) \in M_f \setminus M_b$ gefunden werden kann. Welches Element aus der Menge genommen wird, verändert die Qualität des definierten Ablaufplans nicht, er ist in jedem Fall zulässig, was nun gezeigt wird.

Es genügt, (i) von Def. H (Kap. II.2) für die Zeitpunkte x_{i_1} ($1 \leq l \leq k$) nachzuweisen.

$l = 1$: aus Schritt ② des Algorithmus folgt:

$$a \neq b \in T_1' \Rightarrow \sigma_2(a) \neq \sigma_2(b)$$

$$\text{und somit: } \bigwedge_{1 \leq i \leq m} |\xi_1(x_{i_1})| \leq 1.$$

$l > 1$: aus Schritt ⑥ des Algorithmus folgt:

$$a \in T_1'' \wedge b \in T_1' \Rightarrow \sigma_2(a) \neq \sigma_2(b)$$

zusammen mit Schritt ② induktiv über l :

$$a \neq b \in T_1' \cup T_1'' \Rightarrow \sigma_2(a) \neq \sigma_2(b)$$

$$\text{und somit: } \bigwedge_{1 \leq i \leq m} |\xi_i(x_{i_1})| \leq 1.$$

Die Eigenschaft (ii) von Def. H (Kap. II.2) besitzt $\underline{\sigma}$ wegen $\sigma_1(a_i) = x_i$ ($1 \leq i \leq n$) und der Voraussetzung (ii) des Satzes.

Der Nachweis von (iii) Def. H (Kap. II.2) ergibt sich wie im 1. Schritt des Beweises. Die Gültigkeit von (iv) Def. H (Kap. II.2) erhält man analog zur Überprüfung der Eigenschaft 4. Es folgt die Gleichheit

$$\bigwedge_{1 \leq i \leq n} \bigwedge_{t \in \mathbb{R}^{\geq 0}} \sum_{i=1}^n \mu_1(a_i) \eta_i(t) = \sum_{i=1}^m \mu_1(\xi_i(t))$$

(2. Summe über alle i mit $\xi_i(t) \neq \emptyset$)

aus den Voraussetzungen des Satzes und der Definition von σ_1 direkt wegen der Beziehung

$$\bigwedge_{t \in \mathbb{R}^{\geq 0}} \bigwedge_{1 \leq i \leq n} (\eta_i(t) = 1 \Rightarrow \bigvee_{1 \leq j \leq m} \xi_j(t) = \{a_i\}) \wedge$$

$$\wedge (\eta_i(t) = 0 \Rightarrow \bigwedge_{1 \leq j \leq m} \xi_j(t) \neq \{a_i\})$$

Somit gilt die Abschätzung

$$\bigwedge_{1 \leq i \leq n} \bigwedge_{t \in \mathbb{R}^{\geq 0}} \sum_{i=1}^m \mu_1(\xi_i(t)) \leq 1$$

(2. Summe über alle i mit $\xi_i(t) \neq \emptyset$)

Der konstruierte Ablaufplan ist zulässig. $\square$

Eine Erweiterung dieses Satzes auf p-inhomogene Prozessorsysteme kann durch Modifizierung der Bedingung (i) vorgenommen werden.

Korollar 2 (Kap. III.1)

Sei (G, B) ein Arbeitensystem mit einem p-inhomogenen Prozessorsystem. Dann existiert ein zulässiger Ablaufplan $\underline{\sigma}$ genau dann, wenn es n (Anzahl der Arbeiten in G) Intervallindikatorfunktionen

$$\eta_i : \mathbb{R}^{\geq 0} \rightarrow \{0, 1\}$$

gibt, die folgenden Bedingungen genügen:

$$(i) \quad \begin{array}{c} \wedge \\ 1 \leq l \leq p \end{array} \quad \begin{array}{c} \wedge \\ t \in \mathbb{R}^{\geq 0} \end{array} \quad \sum_{a_i \in T} \eta_i(t) \leq m_l - m_{l-1} \quad (1)$$

(ii) - (iv) wie im vorhergehenden Satz 1 (Kap. III.1)

Bew.:

(i) ist notwendige Bedingung für die Existenz eines zulässigen Ablaufplans. Das folgt unmittelbar aus (i) - (iii) der Def. H (Kap. II.2). Von jedem Prozessortyp l ($1 \leq l \leq p$) existieren nicht mehr als $m_l - m_{l-1}$ Exemplare. Die Bestimmung von $\underline{\sigma}$ aus vorgegebenen η_i ($1 \leq i \leq n$), die den Bedingungen (i) - (iv) des Korollars genügen, erfordert lediglich ein mehrfaches Anwenden des Algorithmus im Beweis des 2. Schrittes des vorigen Satzes auf alle vorhandenen p Prozessortypen. Es sei für $1 \leq i \leq p$

$$x_j^i = \{x_j^i : a_j \in T^{(i)} \wedge x_j^i = \inf \{t \in \mathbb{R}^{\geq 0} : \eta_j(t) = 1\}\}$$

Diese Mengen lassen sich jeweils in einer Folge

$$x_{j_1}^i < x_{j_2}^i < \dots < x_{j_i}^i \quad (1 \leq i \leq p, j_i \leq n)$$

anordnen. Den Elementen dieser Folgen sind analog zum vorhergehenden Satz die Mengen

$$(T_1^i)' = \{a_j \in T^{(i)} : \eta_j(x_{j_1}) = 1 \wedge \bigwedge_{\epsilon > 0} \eta_j(x_{j_1} - \epsilon) = 0\}$$

$$(T_1^i)'' = \{a_j \in T^{(i)} : \eta_j(x_{j_1}) = 1 \wedge \bigvee_{\epsilon > 0} \eta_j(x_{j_1} - \epsilon) = 0\}$$

$$(1 \leq i \leq p, 1 \leq l \leq j_i)$$

zugeordnet. Ein Ablaufplan $\underline{\sigma}$ ist dann gegeben durch die Festlegung

$$\bigwedge_{1 \leq i \leq n} \sigma_1(a_i) = \inf \{t \in \mathbb{R}^{\geq 0} : \eta_i(t) = 1\}$$

und durch rekursive Gewinnung der Komponente σ_2 mit dem erweiterten Algorithmus

$$\textcircled{1} \quad i = 1; l = 1; M_f = \{m_{i-1}+1, m_{i-1}+2, \dots, m_i\}; M_b = \emptyset$$

$$\textcircled{2} \quad a \in (T_1^i)' \Rightarrow \sigma_2(a) \in M_f \setminus M_b, M_b = M_b \cup \{\sigma_2(a)\}$$

$$\textcircled{3} \quad (T_1^i)' \neq \emptyset \Rightarrow \textcircled{2}$$

$$\textcircled{4} \quad l = l + 1$$

$$\textcircled{5} \quad l = j_i + 1 ? \Rightarrow \textcircled{8}$$

$$\textcircled{6} \quad M_b = \{\sigma_2(a) : a \in (T_1^i)''\}$$

$$\textcircled{7} \quad \textcircled{2}$$

$$\textcircled{8} \quad i = i + 1$$

$$\textcircled{9} \quad i = p + 1 ? \Rightarrow \textcircled{11}$$

$$\textcircled{10} \quad \textcircled{2}$$

$$\textcircled{11} \quad \text{end}$$

Dieser ist identisch mit dem von Satz 1 (Kap. III.1), falls das p-Prozessorsystem ein p=1 Prozessorsystem ist. Er erstellt einen zulässigen Ablaufplan, wobei der Beweis dieser Eigenschaft wiederum unmittelbar aus einem p-fachen Anwenden der Schlüsse des vorigen Satzes resultiert. $\square$

Mit diesem Korollar ist eine konstruktive Charakterisierung der Fragestellung $(G, B, \text{ex. } \sigma ?)$ für p-inhomogene Prozessorsysteme gegeben, insbesondere also für homogene. Das Ergebnis läßt sich leicht auf das Problem $(G, B, \text{opt. } \kappa)$ erweitern.

Korollar 3 (Kap. III.1)

Sei (G, B) ein Arbeitensystem mit p-inhomogenem Prozessorsystem. Dann ist das Bestimmen eines κ -optimalen zulässigen Ablaufplans äquivalent mit folgender Optimierungsaufgabe:

Gesucht sind n (Anzahl der Arbeiten in G) Intervallindikatorfunktionen $\eta_i = \mathbb{R}^{\geq 0} \rightarrow \{0, 1\}$ ($1 \leq i \leq n$), die die Zielfunktion

$$\kappa((\inf\{t \in \mathbb{R}^{\geq 0} : \eta_i(t) = 1\}) + \tau_{l_i}(a_i)) \quad 1 \leq i \leq n$$

unter den Nebenbedingungen (i) - (iv) des Korollars 2 (Kap. III.1) minimieren. Der Index l_i gibt den Prozessortyp an, auf dem jedes $a_i \in T$ gerechnet werden kann.

Bew.:

Gemäß Definition der Bewertung von Ablaufplänen (Def. I (Kap. II.2)) stimmt diese mit obiger Zielfunktion überein. Die Nebenbedingungen charakterisieren alle zulässigen Ablaufpläne von (G, B) nach Korollar 2 (Kap. III.1), womit die Lösung der Optimierungsaufgabe einen κ -minimalen Ablaufplan liefert und umgekehrt. $\square$

Die beiden letzten Ergebnisse kennzeichnen zwar scharf die mathematische Struktur der Problemstellungen $(G, B, \text{ex. } \underline{g} \text{ ?})$ und $(G, B, \text{opt. } \kappa)$, sind jedoch nicht geeignet, praktische Auswertungen vorzunehmen. Es fehlen Verfahren zur Lösung von Optimierungsaufgaben, die aus überabzählbar vielen Nebenbedingungen aufgebaut sind und dadurch entstehen, daß (i) und (iv) von Satz 1 (Kap. III.1) für alle $t \in \mathbb{R}^{\geq 0}$ nachzuprüfen sind. Eine zusätzliche Erschwernis kommt über (ii) und (iii) herein. Dort sind nämlich Integralgleichheiten und -Ungleichheiten einzuhalten.

Zusammenfassend haben wir in diesem Abschnitt hergeleitet, daß die Fragestellung $(G, B, \text{ex. } \underline{g} \text{ ?})$ äquivalent mit der Lösbarkeit eines Integralungleichungs-/gleichungssystem ist und das Problem $(G, B, \text{opt. } \kappa)$ eine klassische Optimierungsaufgabe mit Zielfunktion und Nebenbedingungen, die ebenfalls durch Integralgleichungen und -ungleichungen gebildet werden. Im nächsten Absatz werden wir die Zuordnungsprobleme so einschränken, daß die Integrale durch einfache Summen ersetzt werden können und die Nebenbedingungen auf endlich viele zusammenschrumpfen. Wir werden sehen, daß es dann konstruktive mathematische Verfahren für die Beantwortung der Existenzfrage von zulässigen Ablaufplänen gibt und es auch möglich wird, numerisch einen optimalen Ablaufplan $\underline{g}$ für ein gegebenes Arbeitensystem zu finden.

III.2 Die Äquivalenz von $(G, B, \text{ex. } \underline{g} \text{ ?})$ mit der Lösbarkeit eines PB-G/U-Systems (Pseudo-Boolesches Gleichungs-/Ungleichungssystem) [Hammer & Rudeanu 68]

Der Einsatz numerischer Verfahren zur Lösung der in dieser Arbeit gestellten Zuordnungsprobleme setzt auf der Grundlage der bisher abgeleiteten Dualitätsbeziehungen weitere Begriffe voraus.

Def. L (Kap. III.2): Sei σ Ablaufplan eines Arbeitensystems (G, B) . Dann heißen die Mengen

$$T_f(t) = \{a \in T : \sigma_1(a) + \tau_{\sigma_2(a)}(a) \leq t\} \quad \text{und}$$

$$T_b(t) = \{a \in T : UV(a) \subseteq T_f(t) \wedge \tau_f(a) \leq t \wedge \sigma_1(a) > t\}$$

Fertig- und Bereitmengen zu den Zeitpunkten $t \in \mathbb{R}^{\geq 0}$.

$T_f(t)$ beinhaltet alle Arbeiten, die zum Zeitpunkt t bereits beendet sind. Aufgaben $a \in T$, die zum Zeitpunkt t begonnen werden könnten, weil ihre Vorgänger bereits fertig sind, der Freigabezeitpunkt erreicht wurde und deren Bearbeitung aufgrund des Ablaufplans jedoch erst später beginnt, werden in der Menge $T_b(t)$ zusammengefaßt. Intuitiv scheint klar, daß eine Arbeit nur deshalb zur Menge $T_b(t)$ gehört, weil im Zeitpunkt t kein geeigneter Prozessor zur Verfügung steht oder die benötigten Betriebsmittel nicht ausreichen. Andernfalls würde man von einer "vermeidbaren Prozessorlücke" sprechen. Dieses intuitive Verständnis ist jedoch falsch, wie sich im folgenden zeigt. Das strikte Beginnen von Arbeiten immer dann, wenn ein geeigneter Prozessor zur Verfügung steht und die teilbaren Betriebsmittel im Moment ausreichen, d.h. das Nutzen solcher scheinbar vermeidbarer Prozessorlücken, führt im allgemeinen zur einer Verfälschung der Existenzfrage von zulässigen Ablaufplänen.

Def. M (Kap. III.2): Ein zulässiger Ablaufplan σ für das Arbeitensystem (G, B) erzeugt im Intervall $[t_0, t_1]$ ($t_0, t_1 \in \mathbb{R}^{\geq 0}$, $t_0 < t_1$) eine vermeidbare Lücke auf dem Prozessor i ($1 \leq i \leq m$: Anzahl der vorhandenen Prozessoren), wenn:

$$(i) \quad \bigvee_{a \in T_b(t_0)} t_0 + \tau_i(a) \leq \min \{t_1, \sigma_1(a) + \tau_{\sigma_2(a)}(a)\}$$

$$(ii) \quad \begin{array}{c} \wedge \\ t_0 \leq t < t_1 \end{array} \quad \xi_i(t) = \emptyset \quad \vee \quad \xi_i(t) = \{a\}$$

$$(iii) \quad \begin{array}{c} \wedge \\ 1 \leq i \leq s \end{array} \quad t \in [t_0, t_1) \setminus [\sigma_1(a), \sigma_1(a) + \tau_{\sigma_2(a)}(a))$$

$$\sum_{i=1}^m \mu_1(\xi_i(t)) + \mu_1(a) \leq 1$$

σ heißt lückenfrei, wenn auf allen Prozessoren keine vermeidbaren Lücken erzeugt werden.

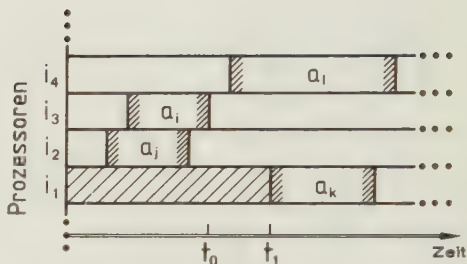
Eine vermeidbare Lücke des Prozessors i ($1 \leq i \leq m$) im Intervall $[t_0, t_1)$ heißt maximal, wenn für alle hinreichend kleinen $\varepsilon \in \mathbb{R}^{\geq 0}$ (nämlich solche mit $\varepsilon \leq t_0$) weder $[t_0 - \varepsilon, t_1)$ noch $[t_0, t_1 + \varepsilon)$ eine vermeidbare Lücke auf dem gleichen Prozessor ist.

Das folgende Beispiel veranschaulicht diesen intuitiv nicht erwarteten Begriff einer vermeidbaren Prozessorlücke im Bild III.2. Auf zwei Eigenschaften sei hier kurz verwiesen. Zum einen gibt es keine vermeidbaren Lücken $[t_0, t_1)$ mit der Eigenschaft

$$t_0 \geq \max \{ \sigma_1(a) : a \in T \}$$

da für solche t_0 gilt: $T_b(t_0) = \emptyset$. Damit ist die Bedingung (i) von Def. M (Kap. III.2) nicht mehr erfüllbar, und es kann insbesondere ab dem Zeitpunkt t_0 , zu dem alle Arbeiten erledigt sind, nicht mehr von vermeidbaren Lücken gesprochen werden. Zum anderen beeinträchtigt ein Vorziehen der Erledigung einer gemäß (i) existierenden Arbeit den sonstigen Ablaufplan nicht. Genau dies kann aber geschehen, wenn wir das intuitive Verständnis einer vermeidbaren Prozessorlücke zugrunde gelegt hätten.

Homogenes Prozessorsystem



(t_0, t_1) : Vermeidbare Lücke des Prozessors i_1 , falls a_k keine Forderungen nach teilbaren Betriebsmitteln stellt.

(t_0, t_1) : Keine vermeidbare Lücke des Prozessors i_1 , falls a_k und a_i ein gemeinsames teilbares Betriebsmittel zu mehr als 100% belegen.

Bild III.2: Vermeidbare Prozessorlücken

$$G_a = (\{a_1, a_2, a_3\}, \phi)$$

$$B_a = (\underline{\tau}, \underline{\mu}, \tau_e \equiv 0, \tau_e, \kappa)$$

$$\underline{\tau}(a_1) = (\infty, 1), \underline{\tau}(a_2) = (\infty, 1), \underline{\tau}(a_3) = (2, \infty)$$

$$\underline{\mu}(a_1) = \frac{1}{3}, \quad \underline{\mu}(a_2) = \frac{2}{3}, \quad \underline{\mu}(a_3) = \frac{2}{3}$$

$$\tau_e(a_1) = 1, \quad \tau_e(a_2) = 2, \quad \tau_e(a_3) = 4$$

P1	a ₃		
P2	a ₁	a ₂	

Listenschedule
(nicht zulässig)

		a ₃	
a ₁	a ₂		

Ablaufplan
(zulässig)

Bild III.3: Einschränkung des Begriffs Listenschedule

Würde nämlich nur überprüft werden, ob zum Zeitpunkt t_0 die Betriebsmittel für eine sonst zur Durchführung bereite Aufgabe zur Verfügung stehen, dann könnte dies dazu führen, daß später gestartete Arbeiten u.U. verzögert werden müssen, weil deren Betriebsmittelanforderungen nicht mehr erfüllt werden können (Bild III.3). Betrachten wir zuerst eine vermeidbare Lücke auf einem Prozessor i gemäß obiger Def. M (Kap. III.2). Es existiert also eine Arbeit $a \in T$ (wegen (i)), die bereits zum Zeitpunkt t_0 auf dem unbelegten Prozessor i (wegen (ii)) begonnen werden kann, da die gewünschten Betriebsmittel vorhanden (wegen (iii)) und auch alle sonstigen Voraussetzungen erfüllt sind. Das Vorziehen des Bearbeitungsbeginns von a ändert die Zulässigkeit des Ablaufs nicht. Betrachte dazu

$$\underline{\sigma}' : T \rightarrow \mathbb{R}^{\geq 0} \times \{1, 2, \dots, m\}$$

$$b \rightsquigarrow \begin{cases} \underline{\sigma}(b), & \text{falls } b \neq a \\ (t_0, i), & \text{falls } b = a. \end{cases}$$

Wegen (i) ist gewährleistet, daß die nachfolgenden Arbeiten $b > a$ durch das Vorziehen von a keine Verzögerungen erfahren. Es gilt nämlich $\sigma_1'(a) + \tau_{\sigma_2'}(a)(a) \leq \sigma_1(a) + \tau_{\sigma_2}(a)(a)$. Im Intervall

$[\sigma_1'(a), \sigma_1'(a) + \tau_{\sigma_2'}(a)(a)]$ kann es aber auch keinen Engpaß

bei den teilbaren Betriebsmitteln gegenüber dem alten Ablaufplan σ geben. Dafür sorgt (iii), so daß $\underline{\sigma}'$ ebenfalls zulässig ist und überdies die Eigenschaften

$$(*) \quad \bigwedge_{a \in T} \sigma_1'(a) \leq \sigma_1(a)$$

$$(2*) \quad \bigwedge_{a \in T} \sigma_1'(a) + \tau_{\sigma_2'}(a)(a) \leq \sigma_1(a) + \tau_{\sigma_2}(a)(a)$$

besitzt, also "besser" als $\underline{\sigma}$ ist. Die Eigenschaft (2*) wäre im allgemeinen falsch, wenn in (i) nur $t_0 + \tau_i(a) \leq t_1$ gefordert worden wäre. Dann könnte nämlich a unter $\underline{\sigma}$ trotz des Vorziehens der Bearbeitung früher fertig sein, nämlich dann, wenn der Prozessor i langsam genug rechnet gegenüber dem sonst benützten Prozessor $\sigma_2(a)$. Ja es könnte sogar soweit kommen, daß durch die vorgezogene Bearbeitung der späteste Beendigungszeitpunkt $\tau_e(a)$ überschritten wird und damit die Zulässigkeit des Ablaufplans verlorengeht.

Ähnliches gilt, wie bereits angedeutet, wenn die Bedingung (iii) der Def. H abgeschwächt wird zu

$$(iii)' \quad \bigwedge_{1 \leq l \leq s} \sum_{i=1}^m \mu_l(\xi_i(t_0)) + \mu_l(a) \leq 1.$$

Das Bild III.3 zeigt, daß das Nutzen derartiger vermeidbarer Lücken analog zu oben ebenfalls zur Überschreitung späterster Beendigungszeitpunkte führen kann. Das ist insofern interessant, als "Listenschedules" [Coffman & Denning 73] genau solche Lücken prinzipiell nützen. Sie arbeiten nämlich mit einer statischen Prioritätsliste der auszuführenden Arbeiten. Ist ein Prozessor frei, so durchsucht er diese und greift die Aufgabe heraus, deren Vorgänger bereits fertig sind, deren Freigabezeitpunkte erreicht wurden, deren Betriebsmittelforderungen derzeit erfüllbar sind und nimmt unter all diesen die mit der höchsten Priorität. In diesem Sinn besitzt das Beispiel von Bild III.3 keinen zulässigen Listenschedule. Der Prozessor 1 würde bereits zum Zeitpunkt $\emptyset$ die Bearbeitung von a_3 aufnehmen und damit das vorhandene Betriebsmittel für die Arbeit a_2 bis zum Zeitpunkt 2 sperren. Hierdurch kann der späteste Beendigungszeitpunkt $\tau_e(a_2) = 2$ nicht eingehalten werden.

Listenschedules sind eine Einschränkung gegenüber dem in dieser Arbeit verwendeten Begriff des Ablaufplans.

Es gibt Zuordnungsprobleme (G,B,ex.o?), für die kein zulässiger Listenschedule existiert, wohl aber ein zulässiger Ablaufplan.

Das liegt offensichtlich an der Verbindung von Forderungen nach teilbaren Betriebsmitteln und dem Einhalten von spätesten Beendigungszeitpunkten. Während das erste bei allen Strategien berücksichtigt wird, die in heute üblichen Mehrprogramm-Rechenanlagen implementiert sind, trifft das zweite nur für Echtzeitdatenverarbeitung zu. Das Beispiel zeigt, daß "Scheduler" in derartigen Betriebssystemen im allgemeinen nicht mit einer statischen Prioritätsliste arbeiten dürfen, sondern den Aufwand einer dynamischen Prioritätsverwaltung treiben müssen, wenn sie nicht von vorneherein zeitkritische Einschränkungen bei der Abwicklung von Aufträgen hinnehmen wollen. Bei Mehrprozessoranlagen wird diese in Verbindung mit prozessor-spezifischen Prioritätsschwellen stehen, bei deren Erreichen erst einzelne Aufträge gerechnet werden dürfen. Hat die dynamische Priorität eines Auftrages diesen Schwellwert nicht erreicht, dann darf selbst bei Erfülltsein aller Anforderungen ein sonst leerlaufender Prozessor die Arbeit nicht beginnen. So unglaublich es klingt: erst die Faulheit (nichts tun, obwohl Arbeit da ist) von Prozessoren ermöglicht Echtzeitbetrieb.

Kehren wir nun zum hier eingeführten Begriff der vermeidbaren Prozessorlücke zurück (Def. M (Kap. III.2)) und zeigen wir, daß dieser Begriff gerechtfertigt ist, d.h. solche Lücken $[t_0, t_1)$ tatsächlich ohne Einschränkung nutzbar sind.

Lemma 4 (Kap. III.2)

Sei $\underline{\sigma}$ ein zulässiger Ablaufplan des Arbeitensystems (G, B) und $[t_0, t_1)$ vermeidbare, maximale Lücke des Prozessors i ($1 \leq i \leq m$). Dann ist t_0 entweder Freigabe- oder Fertigstellungszeitpunkt einer Arbeit $a \in T$, d.h.

$$\begin{array}{c} \diagup \quad \diagdown \\ a \in T \end{array} \tau_f(a) = t_0 \vee \sigma_1(a) + \tau_{\sigma_2(a)}(a) = t_0$$

Bew.:

Sei $[t_0, t_1)$ vermeidbare, maximale Lücke des Prozessors i ($1 \leq i \leq m$). Wegen Def. M (Kap. III.2) existiert ein $a \in T_b(t_0)$ mit den Eigenschaften (i) - (iii).

$$\text{Annahme: } \begin{array}{c} \diagup \quad \diagdown \\ a \in T \end{array} \tau_f(a) \neq t_0 \neq \sigma_1(a) + \tau_{\sigma_2(a)}(a) \quad (*)$$

$$\text{wegen (ii) gilt: } \begin{array}{c} \diagup \quad \diagdown \\ \varepsilon_1 \in \mathbb{R}^{\geq 0} \end{array} \begin{array}{c} \diagup \quad \diagdown \\ 0 \leq \delta \leq \varepsilon_1 \end{array} \xi_i(t_0 - \delta) = \emptyset \quad (2*)$$

$$\text{wegen (*) gilt: } \begin{array}{c} \diagup \quad \diagdown \\ \varepsilon_2 \in \mathbb{R}^{\geq 0} \end{array} \begin{array}{c} \diagup \quad \diagdown \\ 0 \leq \delta \leq \varepsilon_2 \end{array} a \in T_b(t_0 - \delta) \quad (3*)$$

d.h. a gehört bereits vor dem Zeitpunkt t_0 zur Bereitmenge. Aus (*) folgt weiterhin:

$$\begin{array}{c} \diagup \quad \diagdown \\ \varepsilon_3 \in \mathbb{R}^{\geq 0} \end{array} \begin{array}{c} \diagup \quad \diagdown \\ 0 \leq \delta \leq \varepsilon_3 \end{array} \begin{array}{c} \diagup \quad \diagdown \\ t_0 - \delta \neq \sigma_1(a) + \tau_{\sigma_2(a)}(a) \end{array} a \in T$$

$$\Rightarrow \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq l \leq s \end{array} \begin{array}{c} \diagup \quad \diagdown \\ 0 \leq \delta \leq \varepsilon_3 \end{array} \sum_{i=1}^m \mu_1(\xi_i(t_0 - \delta)) \leq \sum_{i=1}^m \mu_1(\xi_i(t_0))$$

Das ergibt zusammen mit (iii)

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq 1 \leq s \quad 0 \leq \delta \leq \epsilon_3 \end{array} \quad \sum_{i=1}^m \mu_1(\xi_i(t_0 - \delta)) + \mu_1(a) \leq 1 \quad (4*)$$

Aus (2*), (3*) und (4*) folgt mit $\epsilon := \min \{\epsilon_1, \epsilon_2, \epsilon_3\}$, daß $[t_0 - \epsilon, t_1)$ eine vermeidbare Lücke des Prozessors i ist. Diese widerspricht der Maximalität von $[t_0, t_1)$. $\square$

Lemma 4 legt die Vermutung nahe, daß bei lückenfreien Ablaufplänen ähnliches für die Startzeitpunkte aller Arbeiten gilt.

Lemma 5 (Kap. III.2)

Sei σ lückenfreier Ablaufplan des Arbeitensystems (G, B) . Dann beginnt die Berechnung jeder Aufgabe $a \in T$ entweder in ihrem Freigabezeitpunkt oder unmittelbar nach Beendigung einer anderen Arbeit, d.h.

$$\begin{array}{c} \diagup \quad \diagdown \\ a \in T \end{array} \tau_f(a) = \sigma_1(a) \vee \begin{array}{c} \diagdown \quad \diagup \\ b \in T \end{array} \sigma_1(b) + \tau_{\sigma_2(b)}(b) = \sigma_1(a)$$

Bew.:

Annahme: es existiert eine Arbeit $a \in T$ mit

$$\sigma_1(a) \neq \tau_f(a) \quad (*)$$

$$\text{und} \quad \begin{array}{c} \diagup \quad \diagdown \\ b \in T \end{array} \sigma_1(a) \neq \sigma_1(b) + \tau_{\sigma_2(b)}(b) \quad (2*)$$

$$\text{Aus (2*)} : \begin{array}{c} \diagup \quad \diagdown \\ \epsilon_1 \in \mathbb{R}^{\geq 0} \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ 0 \leq \delta \leq \epsilon_1 \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ b \in T \end{array} \quad \delta_1(a) - \delta \neq \sigma_1(b) + \tau_{\sigma_2(b)}(b) \quad (3^*)$$

$$\Rightarrow \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq l \leq s \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ 0 \leq \delta \leq \epsilon_1 \end{array} \quad \sum_{i=1}^m \mu_1(\xi_i(\sigma_1(a))) \geq \sum_{i=1}^m \mu_1(\xi_i(\sigma_1(a) - \delta)). \quad (4^*)$$

$$\text{Ebenfalls aus (2*)} : \begin{array}{c} \diagup \quad \diagdown \\ \epsilon_2 \in \mathbb{R}^{\geq 0} \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ 0 \leq \delta \leq \epsilon_2 \end{array} \quad \xi_{\sigma_2(a)}(\sigma_1(a) - \delta) = \emptyset \quad (5^*)$$

Die Gliederungen (4*) und (5*) ergeben zusammen:

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq l \leq s \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ 0 \leq \delta \leq \min\{\epsilon_1, \epsilon_2\} \end{array} \quad \sum_{i=1}^m \mu_1(\xi_i(\sigma_1(a))) \geq \sum_{i=1}^m \mu_1(\xi_i(\sigma_1(a) - \delta)) + \mu_1(a).$$

Da $\underline{g}$ zulässig vorausgesetzt ist:

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq l \leq s \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ 0 \leq \delta \leq \min\{\epsilon_1, \epsilon_2\} \end{array} \quad \sum_{i=1}^m \mu_1(\xi_i(\sigma_1(a) - \delta)) + \mu_1(a) \leq 1 \quad (6^*)$$

$$\text{Aus (*) folgt:} \quad \begin{array}{c} \diagup \quad \diagdown \\ \epsilon_3 \in \mathbb{R}^{\geq 0} \end{array} \quad \tau_f(a) \leq \sigma_1(a) - \epsilon_3 \quad (7^*)$$

Aus (7*) und (3*) folgt für $\epsilon := \min\{\epsilon_1, \epsilon_2, \epsilon_3\}$ die Beziehung $a \in T_b(\sigma_1(a) - \epsilon)$.

Das Intervall $[\sigma_1(a) - \epsilon, \sigma_1(a) + \tau_{\sigma_2(a)}(a)]$ ist damit eine vermeidbare Lücke auf dem Prozessor $\sigma_2(a)$, denn die Eigenschaft

(i) folgt aus (7*), (ii) aus (5*) und (iii) aus (6*). Das ist ein Widerspruch zur Lückenfreiheit des Ablaufplans σ . $\square$

Als Konsequenz von Lemma 5 (Kap. III.2) ergibt sich eine weitere Verschärfung für die Startzeitpunkte der Arbeiten bei lückenfreien, zulässigen Ablaufplänen unter der Voraussetzung zusätzlicher Angaben über die Rechenzeiten und Freigabezeitpunkte.

Korollar 6 (Kap. III.2)

Sei (G, B) ein Arbeitensystem mit natürlichen Rechenzeiten und Freigabezeitpunkten. Dann erfüllt jeder zulässige, lückenfreie Ablaufplan σ :

$$\bigwedge_{a \in T} \sigma_1(a) \in \mathbb{N}_0 \quad \mathbb{N}_0 = \mathbb{N} \cup \{0\}$$

Bew.:

Unter Zuhilfenahme von Lemma 5 (Kap. III.2) folgt die obige Eigenschaft induktiv über die aufsteigende Folge der Startzeitpunkte $\{ \sigma_1(a) : a \in T \}$. $\square$

Die Einschränkung auf natürliche Rechenzeiten und Freigabezeitpunkte erzeugt deshalb bei den Arbeitsfunktionen

$$\eta_i : \mathbb{R}^{\geq 0} \rightarrow \{0, 1\} \quad (1 \leq i \leq n)$$

nur noch auf den Einheitsintervallgrenzen Änderungen ihrer Funktionswerte. Sie können somit unter diesen Voraussetzungen vollständig durch die Werte auf $\mathbb{N}_0$ beschrieben werden, die spätestens ab $\lfloor \tau_e(a_i) \rfloor$ ($1 \leq i \leq n$) alle den Wert $\emptyset$ annehmen müssen. Es ist offensichtlich, daß dann auch alle Beendigungszeitpunkte $\tau_e(a)$ ($a \in T$) ohne Einschränkung als natürlich vorausgesetzt werden können.

Der Übergang

$$\tau_e(a) \rightarrow \lfloor \tau_e(a) \rfloor = \max \{n \in \mathbb{N}_0 : n \leq \tau_e(a)\}$$

ändert an der Problemstellung wegen Korollar 6 (Kap. III.2) nichts.

Definiert man T_e als einen Zeitpunkt, zu dem spätestens alle Arbeiten beendet sein müssen, z.B.

$$T_e = \max \{ \tau_e(a) : a \in T \}$$

dann kann jede der Arbeitsfunktionen η_i ($1 \leq i \leq n$) durch T_e Boole Variable η_{ij} ($1 \leq j \leq T_e$) beschrieben werden, die nach folgender Vorschrift die Beziehung zu η_i herstellen.

$$\eta_{ij} = 1 \Leftrightarrow \eta_i(j-1) = 1 \quad (1 \leq i \leq n,$$

$$\eta_{ij} = 0 \Leftrightarrow \eta_i(j-1) = 0 \quad 1 \leq j \leq T_e)$$

Diese Vereinfachung führt dazu, das eingeschränkte Problem $(G, B, ex, \underline{0}?)$ einfach und mathematisch konstruktiv charakterisieren zu können. Hierzu wird die Allgemeinheit lückenfreier Ablaufpläne benötigt.

Lemma 7 (Kap. III.2)

Sei (G, B) ein Arbeitensystem. Dann existiert ein zulässiger Ablaufplan genau dann, wenn ein zulässiger, lückenfreier für (G, B) existiert.

Bew.:

Intuitiv ist klar, daß durch "geeignetes Zusammenschieben" eines zulässigen, aber nicht lückenfreien Ablaufplans ein lückenfreier entsteht.

Formal zeigen wir nachfolgend, daß durch rekursive Änderung einer einzigen Stelle im Ablaufplan jeweils eine vermeidbare Lücke verschwindet, ohne seine Zulässigkeit zu verletzen. Nur endlich viele Rekursionen sind nötig, um damit den gesamten Ablaufplan lückenfrei zu bekommen. Das verwendete Verfahren ist konstruktiv. Sei $[t_0, t_1)$ eine vermeidbare, maximale Lücke des Prozessors i , die so gewählt wurde, daß t_0 minimal ist, d.h. alle anderen solchen $[t_0', t_1')$ haben die Eigenschaft $t_0 \leq t_0'$.

Wegen Def. M (Kap. III.2) existiert eine Arbeit a , die bereits in dieser Lücke gerechnet werden kann, ohne den restlichen Ablaufplan zu beeinflussen. Die Änderung

$$\sigma' : T \rightarrow \mathbb{R}^{\geq 0} \times \{1, 2, \dots, m\}$$

$$b \rightsquigarrow \begin{cases} \sigma(b), & \text{falls } b \neq a \\ (t_0, i), & \text{falls } b = a \end{cases}$$

ändert nichts an der Zulässigkeit und erzeugt keine vermeidbaren Lücken $[t_0', t_1')$ mit $t_0' < t_0$.

Da für alle $t \geq t_0$ $a \notin T_b(t)$, muß nach einem maximal n -fachen Anwenden dieser Ersetzung $T_b(t)$ für alle $t \geq t_0$ die leere Menge geworden sein. Spätestens dann ist der Ablaufplan lückenfrei. $\square$

Um die Frage der Existenz eines zulässigen Ablaufplans zu beantworten, brauchen wegen Lemma 7 (Kap. III.2) nur die lückenfreien betrachtet werden. Diese Einschränkung ist nicht zulässig beim Bestimmen eines κ -optimalen Ablaufs mit allgemeiner Kostenfunktion κ , wie wir im nächsten Abschnitt sehen werden.

Zunächst aber eine algebraische Charakterisierung der Existenzfrage, die wegen der Allgemeinheit von lückenfreien $\underline{g}$ hergeleitet werden kann.

Satz 8 (Kap. III.2)

Sei (G, B) ein p -inhomogenes Arbeitensystem mit Einheitsbearbeitungszeiten und natürlichen Freigabezeitpunkten. Dann existiert ein zulässiger Ablaufplan $\underline{g}$ genau dann, wenn es nT_e $(T_e = \max \{\tau_e(a) : a \in T\})$ Boolesche Variablen η_{ij} $(1 \leq i \leq n$: Anzahl der Arbeiten; $1 \leq j \leq T_e)$ gibt, die folgenden Bedingungen genügen:

$$(i) \quad \bigwedge_{1 \leq k \leq p} \bigwedge_{1 \leq j \leq T_e} \sum_{a_i \in T^{(k)}} \eta_{ij} \leq m_k - m_{k-1}$$

$$(ii) \quad \bigwedge_{1 \leq i \leq n} \sum_{j=1}^{T_e} \eta_{ij} = 1 = \sum_{j=\tau_f(a_i)+1}^{T_e} \eta_{ij}$$

$$(iii) \quad \bigwedge_{a_i, a_j \in T} (a_i < a_j \Rightarrow \sum_{k=1}^{T_e} k \eta_{ik} < \sum_{k=1}^{T_e} k \eta_{jk})$$

$$(iv) \quad \bigwedge_{1 \leq l \leq s} \bigwedge_{1 \leq j \leq T_e} \sum_{i=1}^n \mu_l(a_i) \eta_{ij} \leq 1$$

Bew.:

1. Schritt:

Seien $\eta_{ij} \in \{0, 1\}$ $(1 \leq i \leq n, 1 \leq j \leq T_e)$ Lösungen des Gleichungs-/Ungleichungssystems (i) - (iv). Die Abbildungen

$$\begin{array}{c}
 \wedge \\
 1 \leq i \leq n
 \end{array}
 \quad \eta_i : \mathbb{R}^{\geq 0} \rightarrow \{0,1\}$$

$$(*) \quad t \mapsto \begin{cases} \eta_{ij}, & \text{falls } j-1 \leq t < j \wedge t < T_e \\ 0 & \text{sonst.} \end{cases}$$

sind Intervallindikatorfunktionen. Betrachte für ein $1 \leq k \leq p$ (p : Anzahl der Prozessortypen) alle η_i mit $a_i \in T^{(k)}$ und sei $t \in \mathbb{R}^{\geq 0}$.

1. Fall: $t \geq T_e$

$$(*) \Rightarrow \eta_i(t) = 0 \Rightarrow \sum_{a_i \in T^{(k)}} \eta_i(t) = 0 \leq m_k - m_{k-1}$$

2. Fall: $t < T_e$

$$(*) \Rightarrow \eta_i(t) = \eta_{ij} \quad \text{mit } j = \lceil t \rceil \quad (\lceil t \rceil := \min\{n \in \mathbb{N} : n \geq t\})$$

$$\Rightarrow \sum_{a_i \in T^{(k)}} \eta_i(t) = \sum_{a_i \in T^{(k)}} \eta_{ij} \leq m_k - m_{k-1}$$

gemäß Voraussetzung (i).

$$\text{also insgesamt: } \begin{array}{c} \wedge \\ 1 \leq k \leq p \end{array} \quad \begin{array}{c} \wedge \\ t \in \mathbb{R}^{\geq 0} \end{array} \quad \sum_{a_i \in T^{(k)}} \eta_i(t) \leq m_k - m_{k-1}$$

Das ist die Bedingung (i) des Dualitätssatzes 1 (Kap. III.1).

Sei nun $1 \leq i \leq n$. Wegen (*) und Gleichung (ii) gilt:

$$\int_0^{T_e} \eta_i(t) dt = 1 = \int_{\tau_f(a_i)}^{\tau_e(a_i)} \eta_i(t) dt$$

Das ist die Bedingung (ii) des Dualitätssatzes 1 (Kap. III.1).

Wegen der Gleichung (ii) sind alle Indikatorfunktionen (*) auf genau einem Einheitsintervall identisch 1 und verschwinden sonst. Mit einfachen geometrischen Überlegungen (analog Bild III.1) erhält man deshalb:

$$\int_0^{\infty} t \eta_i(t) dt = \sum_{k=1}^{T_e} k \eta_{ik} - \frac{1}{2}$$

Die Ungleichung (iii) der Voraussetzung liefert somit für alle Arbeiten $a_i, a_j \in T$ mit $a_i < a_j$:

$$1 + \int_0^{\infty} t \eta_i(t) dt \leq \int_0^{\infty} t \eta_j(t) dt$$

oder umgekehrt:

$$\int_0^{\infty} t (\eta_j(t) - \eta_i(t)) dt \leq 1.$$

Das ist die Bedingung (iii) des Dualitätssatzes 1 (Kap. III.1), denn es wurde zusätzlich $\tau(a_i) = \tau(a_j) = 1$ vorausgesetzt.

Sei nun $1 \leq l \leq s$ (s : Anzahl der Betriebsmittel) und $t \in \mathbb{R}^{\geq 0}$

1. Fall: $t \geq T_e$

$$(*) \Rightarrow \bigwedge_{1 \leq i \leq n} \eta_i(t) = 0 \Rightarrow \sum_{i=1}^n \mu_l(a_i) \eta_i(t) = 0 \leq 1$$

2. Fall: $t < T_e$

$$(*) \Rightarrow \bigwedge_{1 \leq i \leq n} \eta_i(t) = \eta_{ij} \text{ mit } j = \lceil t \rceil$$

$$\Rightarrow \sum_{i=1}^n \mu_l(a_i) \eta_i(t) = \sum_{i=1}^n \mu_l(a_i) \eta_{ij} \leq 1 \text{ nach Voraussetzung (iv)}$$

Damit ist auch die Bedingung (iv) des Dualitätssatzes 1 (Kap. III.1) nachgewiesen. Die Existenz eines zulässigen Ablaufplans ist sichergestellt.

2. Schritt:

Sei σ zulässiger Ablaufplan des p-inhomogenen Arbeitensystems (G,B). Wegen Korollar 6 (Kap. III.2) kann ohne Einschränkung vorausgesetzt werden, daß für alle $a \in T$ $\sigma_1(a) \in \mathbb{N}_0$ gilt.

Der Dualitätssatz 1 (Kap. III.1) gewährleistet, daß seine Arbeitsfunktionen η_i ($1 \leq i \leq n$: Anzahl der Arbeiten) die dort formulierten Bedingungen (i) - (iv) erfüllen. Wegen der Einheitsbearbeitungszeiten können Boole Variable

$$\eta_{ij} = \eta_i(j-1) \quad (1 \leq i \leq n, 1 \leq j \leq T_e)$$

definiert werden, die den Bedingungen (i) - (iv) dieses Satzes genügen, wie im folgenden gezeigt wird.

Die Ungleichung (i) ist in Bedingung (i) des Dualitätssatzes explizit enthalten, da $\eta_{ij} = \eta_i(t = j - 1)$.

Die Gleichung (ii) ergibt sich unmittelbar aus der Bedingung (ii) des Dualitätssatzes, da für Einheitsbearbeitungszeiten gilt:

$$\int_0^{\infty} \eta_i(t) dt = \sum_{i=1}^{T_e} \eta_{ij}$$

Die Vorgängerungleichung (iii) kann analog zum 1. Schritt des Beweises unmittelbar aus der Integralbeziehung (iii) des Satzes 1 (Kap. III.1) abgeleitet werden.

Das Einhalten der Betriebsmittelbeschränkungen (iv) ist wiederum ein Spezialfall des Kriteriums (iv) im Dualitätssatz.

Die Existenz eines zulässigen Ablaufplans ist also der Nachweis für die Lösbarkeit des Gleichungs-/Ungleichungssystems, das dem Arbeitensystem (G,B) zugeordnet ist und umgekehrt. $\square$

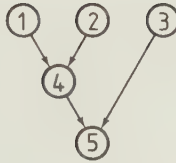
Solche Systeme heißen Pseudo-Boole-Gleichungs-/Ungleichungssysteme (PB-G/U-Systeme) und zeichnen sich dadurch aus, daß sie durch Funktionen der Form

$$f : \{0,1\}^n \rightarrow \mathbb{R}$$

gebildet werden. Ist f eine Pseudo-Boole Funktion, dann nennt man

$f = 0$	eine Pseudo-Boole Gleichung
$f \text{ op } 0$	eine Pseudo-Boole Ungleichung
	(op: $> < \geq \leq$)

In diesem Sinn sind die Bedingungen (i) - (iv) des vorigen Satzes alle entweder Pseudo-Boole Gleichungen oder Pseudo-Boole Ungleichungen. Die Theorie solcher Funktionen ist ausführlich in [Hammer & Rudeanu 68] beschrieben, die insbesondere Lösungsalgorithmen für PB-G/U-Systeme einschließt. Zwei von ihnen werden in einem späteren Kapitel bzgl. der Anwendung auf das Lösen von Zuordnungsproblemen näher behandelt. Die Fragestellung (G,B, ex. $\underline{g}$?) kann unter den Einschränkungen von Satz 8 (Kap. III.2) (Einarbeitungszeiten und natürliche Freigabezeitpunkte) mit beliebigen Arbeitsabhängigkeiten und Betriebsmittelforderung eindeutig für p -inhomogene Prozessorsysteme - und damit auch für homogene - entschieden werden. Das PB-G/U-System, das dieser Fragestellung gemäß Satz 8 (Kap. III.2) äquivalent ist, das sogenannte adjungierte PB-G/U-System, zeichnet sich sogar durch Linearität aus, so daß die Lösungsalgorithmen besonders einfach ablaufen können.



Homogenes Prozessorsystem $M=2$

Einheitsbearbeitungszeiten

keine teilbaren Betriebsmittel

$$\tau_t \equiv 0$$

$$\tau_e \equiv 4$$

Bild III.4: Beispiel 2 zu $(G, B, ex, \sigma?)$

$$\eta_{51} + \eta_{52} + \eta_{53} + \eta_{54} = 1$$

$$\eta_{11} + \eta_{12} + \eta_{13} + \eta_{14} = 1$$

$$\eta_{21} + \eta_{22} + \eta_{23} + \eta_{24} = 1$$

$$\eta_{31} + \eta_{32} + \eta_{33} + \eta_{34} = 1$$

$$\eta_{41} + \eta_{42} + \eta_{43} + \eta_{44} = 1$$

$$\bar{\eta}_{11} + \bar{\eta}_{21} + \bar{\eta}_{31} + \bar{\eta}_{41} + \bar{\eta}_{51} \geq 3$$

$$\bar{\eta}_{12} + \bar{\eta}_{22} + \bar{\eta}_{32} + \bar{\eta}_{42} + \bar{\eta}_{52} \geq 3$$

$$\bar{\eta}_{13} + \bar{\eta}_{23} + \bar{\eta}_{33} + \bar{\eta}_{43} + \bar{\eta}_{53} \geq 3$$

$$\bar{\eta}_{14} + \bar{\eta}_{24} + \bar{\eta}_{34} + \bar{\eta}_{44} + \bar{\eta}_{54} \geq 3$$

$$4\eta_{44} + 4\bar{\eta}_{14} + 3\eta_{43} + 3\bar{\eta}_{13} + 2\eta_{42} + 2\bar{\eta}_{12} + \eta_{41} + \bar{\eta}_{11} \geq 11$$

$$4\eta_{44} + 4\bar{\eta}_{24} + 3\eta_{43} + 3\bar{\eta}_{23} + 2\eta_{42} + 2\bar{\eta}_{22} + \eta_{41} + \bar{\eta}_{21} \geq 11$$

$$4\eta_{54} + 4\bar{\eta}_{34} + 3\eta_{53} + 3\bar{\eta}_{33} + 2\eta_{52} + 2\bar{\eta}_{32} + \eta_{51} + \bar{\eta}_{31} \geq 11$$

$$4\eta_{54} + 4\bar{\eta}_{44} + 3\eta_{53} + 3\bar{\eta}_{43} + 2\eta_{52} + 2\bar{\eta}_{42} + \eta_{51} + \bar{\eta}_{41} \geq 11$$

PB-G/U-System zum Beispiel 2 (Bild III.4)

Die Theorie Pseudo-Boolescher Funktionen ermöglicht darüber hinaus die Bestimmung des Minimums einer nicht notwendig linearen PB-Funktion unter beliebigen Pseudo-Boole Nebenbedingungen. Das versetzt uns im nächsten Abschnitt in die Lage, auch das Zuordnungsproblem $(G, B, \text{opt}_\kappa)$ in großer Breite zu beantworten. Eine wirkungsvolle Anwendung erfuhr die Theorie Pseudo-Boolescher Funktionen bei der Bestimmung minimaler Überdeckungen von Matrizen, was z.B. bei der "Ungarischen Methode" zur Lösung des Transportproblems für Theorie und Praxis bedeutungsvoll wurde (z.B. [Schmid 78]). Auch das Einfärben von Landkarten mit z.B. vier Farben ist damit möglich [Hammer & Rudeanu 68].

Abschließend wird für das einfache Zuordnungsproblem von Bild III.4 das adjungierte PB-G/U-System bestimmt und die Ergebnisse vorgestellt, die unter Benutzung eines solchen Algorithmus' [Hammer & Rudeanu 68] erzielt werden konnten. Das verwendete Lösungsverfahren wurde auf einer CYBER172-Rechenanlage der Firma CDC implementiert und berechnet alle 13 möglichen Ablaufpläne in insgesamt 16 sec, wobei das Abschalten umfangreicher Kontrolldrucke die Rechenzeit um den Faktor 1/2 reduzieren würde. Es wurden 20 Boole Variable benötigt, die einen binären Baum mit 2^{20} Pfaden aufspannen. Um alle 13 Lösungen zu bestimmen, durchsuchte der Algorithmus nur 101 Pfade.

III.3 Die Äquivalenz von $(G, B, \text{opt}_\kappa)$ mit der Lösbarkeit einer Pseudo-Boole Optimierungsaufgabe unter Pseudo-Boole Nebenbedingungen [Hammer & Rudeanu 68]

Im Falle der Existenz von zulässigen Ablaufplänen für ein Arbeitensystem (G, B) interessiert die Bestimmung eines optimalen Ablaufs. Das ist einer, der die im Bearbeitungssystem spezifizierte Kostenfunktion κ mit den Werten der Fertigstellungszeitpunkte minimiert (Def. I Kap. II.2). Das Zuordnungsproblem $(G, B, \text{opt}_\kappa)$ kann wegen der Endlichkeit zweistufig gelöst werden.

Im ersten Schritt werden alle zulässigen Ablaufpläne bestimmt, aus denen anschließend der minimal bewertete ausgewählt wird. Wir werden in diesem Abschnitt sehen, daß es genügt, sich wiederum auf lückenfreie Abläufe zu beschränken, wenn an die Kostenfunktion Bedingungen geknüpft werden, wie sie beispielsweise von der gewichteten mittleren Flußzeit erbracht werden und damit natürlich auch von der Gesamtlänge eines Ablaufplans. Der ableitbare Zuordnungsalgorithmus verknüpft sinnvollerweise beide Schritte miteinander, indem aus der eingeschränkten Menge Σ der zu untersuchenden zulässigen Ablaufpläne erstmal ein beliebiger bestimmt wird, um mit seiner Bewertung Σ erneut einzuschränken, usw., bis schließlich die Existenz eines besser bewerteten ausgeschlossen ist. Der verwendete Algorithmus löst Pseudo-Boole-Optimierungsaufgaben. Sein Einsatz kann nur erfolgen, wenn weitere Aussagen über zulässige Ablaufpläne gewonnen und die betrachteten Kostenfunktionen κ eingeschränkt werden.

Lemma 9 (Kap. III.3)

Sei (G, B) ein p -inhomogenes Arbeitensystem mit Einheitsbearbeitungszeiten und natürlichen Freigabezeitpunkten. Alle Lösungen η_{ij} ($1 \leq i \leq n$: Anzahl der Arbeiten; $1 \leq j \leq T_e$: spätest möglicher Beendigungszeitpunkt aller Arbeiten) des adjungierten PB-G/U-Systems (gemäß Satz 8 (Kap. III.2)) führen genau zu den zulässigen Ablaufplänen $\underline{\sigma}$ mit der Eigenschaft

$$\bigwedge_{1 \leq i \leq n} \sigma_1(a_i) \in \mathbb{N}_0$$

Bew.:

Einer Lösung des PB-G/U-Systems können wie im Beweis von Satz 8 (Kap. III.2) n Intervallindikatorfunktionen η_i ($1 \leq i \leq n$) zugeordnet werden, die Einheitsintervalle anzeigen und den Dualitätssatz anwendbar machen (Satz 1 (Kap. III.1)). In seinem Beweis wird aus den η_i ein Ablaufplan σ hergeleitet, für dessen erste Komponente gilt:

$$\bigwedge_{1 \leq i \leq n} \sigma_1(a_i) = \inf \{t \in \mathbb{R}^{\geq 0} : \eta_i(t) = 1\}$$

Aus dem vorher genannten folgt damit die eine Richtung des Lemmas. Sind umgekehrt die Startzeitpunkte der Bearbeitung jeder Aufgabe gemäß einem Ablaufplan σ natürliche Zahlen, dann können die Boole Variablen η_{ij} unmittelbar aus den Arbeitsfunktionen σ_i ($1 \leq i \leq n$) abgelesen werden. Diese η_{ij} lösen das adjungierte PB-G/U-System (2. Schritt des Beweises von Satz 8 (Kap. III.2)). □

Von Korollar 6 (Kap. III.2) wissen wir, daß lückenfreie, zulässige Ablaufpläne diese Eigenschaften besitzen, weswegen die Existenzfrage (G,B,ex. $\underline{\sigma}$?) über das adjungierte PB-G/U-System entscheidbar war. Nun wissen wir, daß alle lückenfreien, zulässigen Ablaufpläne auch in den Lösungen des adjungierten PB-G/U-Systems auftauchen, was wir zur Bestimmung eines optimalen Ablaufs bei einer bestimmten Klasse von Kostenfunktionen verwenden können.

Def. N(Kap. III.3): Sei $\kappa: (\mathbb{R}^{\geq 0})^n \rightarrow \mathbb{R}$ Kostenfunktion eines Arbeitensystems G,B). Dann heißt κ isoton, falls

$$\bigwedge_{1 \leq i \leq n} \bigwedge_{x \in (\mathbb{R}^{\geq 0})^n} \bigwedge_{\varepsilon > 0}$$

$$\kappa(x_1, x_2, \dots, x_n) \leq \kappa(x_1, x_2, \dots, x_i + \varepsilon, \dots, x_n)$$

κ heißt streng isoton, wenn die strikte Ungleichung gilt.

Die gewichtete mittlere Flußzeit

$$\kappa : (\mathbb{R}^{\geq 0})^n \rightarrow \mathbb{R}$$

$$(x_1, x_2, \dots, x_n) \rightsquigarrow \frac{1}{n} \sum_{i=1}^n g(a_i) x_i \quad g(a_i) \in \mathbb{R} \quad (1 \leq i \leq n)$$

ist offensichtlich genau dann isoton, wenn alle $g(a_i) \geq 0$ ($1 \leq i \leq n$) und genau dann streng isoton, wenn sogar alle $g(a_i) > 0$ sind ($1 \leq i \leq n$). Deshalb ist die Gesamtlänge eines Ablaufplans nicht streng isoton im Gegensatz zur mittleren Flußzeit. Beide Kostenfunktionen sind Spezialfälle der mittleren gewichteten Flußzeit. Im ersten Fall gilt:

$$\begin{array}{ccc} \text{V} & g(a_{i_0}) = 1 \wedge & \text{^} \\ 1 \leq i_0 \leq n & & 1 \leq i \neq i_0 \leq n \end{array} \quad g(a_i) = 0$$

im zweiten: $\text{^} \quad g(a_i) = 1$
 $1 \leq i \leq n$

Lemma 10 (Kap. III.3)

Sei κ isotone Kostenfunktion eines Arbeitensystems (G, B) und die Menge der zulässigen Ablaufpläne nicht leer. Dann existiert ein optimaler Ablaufplan, der sogar lückenfrei ist. Ist κ streng isoton und das Prozessorsystem p -inhomogen, dann sind alle optimalen Ablaufpläne lückenfrei.

Bew.:

Sei κ isoton und $\underline{\sigma}$ ein optimaler, nicht lückenfreier Ablaufplan. Im Beweis von Lemma 7 (Kap. III.2) wurde aus $\underline{\sigma}$ ein lückenfreier, ebenfalls zulässiger Ablaufplan $\underline{\sigma}'$ bestimmt mit der Eigenschaft:

$$\text{^} \quad \sigma_1'(a) + \tau_{\sigma_2'}(a) \leq \sigma_1(a) + \tau_{\sigma_2}(a) \quad a \in T$$

Ein iteriertes Anwenden der Isotonieeigenschaft von κ auf die einzelnen Komponenten liefert:

$$B(\sigma') \neq B(\sigma)$$

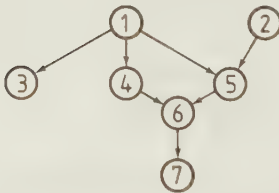
Da $\underline{\sigma}$ optimal war, gilt sogar die Gleichheit. Der lückenfreie Ablaufplan $\underline{\sigma}'$ ist ebenfalls optimal.

Wir hatten $\underline{\sigma}$ nicht lückenfrei vorausgesetzt. Im p-inhomogenen Prozessorsystem ist die Bearbeitungszeit jeder Aufgabe auf allen Prozessoren eines Typs gleich. Durch Nützen von vermeidbaren Lücken erhält man deswegen sogar

$$\bigwedge_{a \in T} \sigma_1'(a) + \tau_{\sigma_2'}(a) < \sigma_1(a) + \tau_{\sigma_2}(a)$$

Dies hat bei streng isotonen Kostenfunktionen die Konsequenz einer besseren Bewertung von σ' , also $B(\sigma') < B(\sigma)$. Unter dieser Voraussetzung müssen optimale Ablaufpläne notwendig lückenfrei sein. $\square$

Ein einfaches Beispiel (Bild III.5) zeigt, daß bei isotonen Kostenfunktionen nicht lückenfreie Abläufe trotzdem optimal sein können und im Fall nicht isotoner Kostenfunktionen sogar keine optimalen lückenfreien $\underline{\sigma}$ existieren müssen.



Homogenes Prozessorsystem $M=2$
 Einheitsbearbeitungszeiten
 keine Betriebsmittelforderungen
 kein vorgeschriebenes Bearbeitungsintervall.

Optimaler Ablaufplan
 bzgl. Gesamtlänge (Isoton)

P1	1	4	6	3
P2	2	5		7

↑ Vermeidbare
 Prozessorlücke

Optimaler Ablaufplan
 bzgl. $\kappa(x_1, x_2, \dots, x_7) = |x_1 - \frac{3}{2}|$
 (nicht Isoton)

P1	2		4	6	7
P2		1	5	3	

↑ Vermeidbare Prozessorlücke

Bild III.5: Eigenschaften κ -optimaler Abläufe

Satz 11 (Kap. III.3)

Sei (G, B) ein p -inhomogenes Arbeitensystem mit Einheitsbearbeitungszeiten und natürlichen Freigabezeitpunkten. Falls κ isoton, so ist die Bestimmung eines κ -minimalen, zulässigen Ablaufplans äquivalent mit dem Lösen einer Pseudo-Boole Optimierungsaufgabe über den Boole Variablen η_{ij} ($1 \leq i \leq n$: Anzahl der Arbeiten; $1 \leq j \leq T_e$: spätester Beendigungszeitpunkt aller Arbeiten) mit der

$$\text{Zielfunktion: } \kappa \left(\sum_{j=1}^{T_e} j \eta_{1j}, \sum_{j=1}^{T_e} j \eta_{2j}, \dots, \sum_{j=1}^{T_e} j \eta_{nj} \right) = \min \quad \text{und den}$$

Nebenbedingungen wie im Satz 8 (Kap. III.2).

Bew.:

Lösungen η_{ij} ($1 \leq i \leq n$; $1 \leq j \leq T_e$) der Nebenbedingungen umfassen nach Lemma 9 alle lückenfreien Ablaufpläne, deren Betrachtung nach Lemma 10 genügt, um bei isotonen Kostenfunktionen κ den optimalen zu bestimmen. Für die Beendigungszeitpunkte der Arbeiten gemäß so erhaltenem zulässigem $\underline{\sigma}$ gilt wegen der Einheitsbearbeitungszeit

$$\bigwedge_{a_i \in T} \sigma_1(a_i) + \tau_{\sigma_2(a_i)}(a_i) = \sum_{j=1}^{T_e} j \eta_{ij}$$

Hieraus folgt für deren Bewertung:

$$B(\underline{\sigma}) = \kappa \left(\left(\sum_{j=1}^{T_e} j \eta_{ij} \right)_{1 \leq i \leq n} \right) \quad \square$$

Die Isotonie von κ ist eine notwendige Voraussetzung für die Gültigkeit dieses Satzes. Betrachte hierzu nochmals das Beispiel von Bild III.5 mit der Kostenfunktion

$$\kappa : (\mathbb{R}^{\geq 0})^n \rightarrow \mathbb{R}$$

$$(x_1, x_2, \dots, x_n) \rightsquigarrow |x_1 - \frac{3}{2}|$$

Der optimale Ablaufplan erzwingt eine Bearbeitung der Aufgabe 1 im Intervall $[1/2, 3/2)$ und besitzt damit nicht die Eigenschaft

$$\bigwedge_{a \in T} \sigma_1(a) \in \mathbb{N}_0.$$

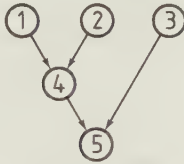
Jedoch werden nur solche Abläufe von den Pseudo-Boole Nebenbedingungen des Satzes erfaßt. Mit den bisherigen Aussagen läßt sich eine mengentheoretische Klassifikation zulässiger Ablaufpläne einrichten, wie sie im Bild III.8 dargestellt ist.

Die Lösung der PB-Optimierungsaufgabe von Satz 11 kann z.B. auf der Basis des Algorithmus von FORTET [Fortet 60] erreicht werden, der im letzten Kapitel dieser Arbeit vorgestellt wird. Für die gängigen Kostenfunktionen "Länge des Ablaufs", "mittlere gewichtete und ungewichtete Flußzeit" sind die zugehörigen Zielfunktionen linear und von der allgemeinen Gestalt

$$\sum_{i=1}^n g(a_i) \left(\sum_{j=1}^{T_e} j \eta_{ij} \right) = \min$$

Diese Optimierungsprobleme können auch mit dem einfacheren Algorithmus von HAMMER und RUDEANU [Hammer & Rudeanu 68] gelöst werden. Ein Anwendungsbeispiel hierfür ist das Zuordnungsproblem von Bild III.4, von dem wir bereits wissen, daß es zulässige Ablaufpläne besitzt. Die Bestimmung eines solchen mit minimaler Länge konnte mit wenig Aufwand vorgenommen werden.

Der Zuordnungsalgorithmus benötigte 20 Boole Variablen und wußte im 8. durchsuchten Pfad des binären Baumes der Tiefe 20 (2^{20} mögliche Pfade), daß die optimale Lösung bereits gefunden ist. Die Rechenzeit hierfür betrug 2 sec. Der ermittelte Ablaufplan ist im Gantt-Diagramm des Bildes III.6 festgehalten.



Homogenes Prozessorsystem $M=2$
 Einheitsbearbeitungszeiten
 keine teilbaren Betriebsmittel
 $\tau_i \equiv 0$ $\tau_e \equiv 4$

P1	1	4	5
P2	2	3	

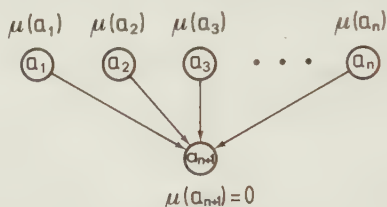
Kürzester Ablaufplan
 bzgl. Gesamtlänge

Bild III.6: Pseudo-Boole Algorithmus löst (G, B, opt_K)

Über den Interpolationssatz für Pseudo-Boole Funktionen [Kap. 1: Hammer & Rudeanu 68] kann jede beliebige isotone Kostenfunktion κ als Zielfunktion der Optimierungsaufgabe erfaßt werden. In Abhängigkeit von ihrer Linearität kommen dann die verschiedenen Lösungsverfahren in Betracht, da in jedem Fall die Nebenbedingungen linear sind und damit die Zielfunktion entscheidet, ob das Problem linear oder nicht linear ist.

Die Aussage des Satzes 11 (Kap. III.2) ist auch auf NP-vollständige Zuordnungsprobleme anwendbar, weswegen Schranken über die Komplexität des Lösungsalgorithmus schwer angebbbar sein werden. Unabhängige Aufgaben, die mit 3 Prozessoren und Einheitsbearbeitungszeiten unter Verwendung eines einzigen teilbaren Betriebsmittels gesamtzeitoptimal abgewickelt werden sollen, stellen nach [Garey & Johnson 75] ein NP-vollständiges Problem dar.

Das kann äquivalent in eine lineare PB-Optimierungsaufgabe umgewandelt werden, in der die n Arbeiten um eine zusätzliche Aufgabe a_{n+1} erweitert werden, die keine Betriebsmittelforderungen stellt und ein Zeitintervall zur Bearbeitung braucht (Bild III.7). a_{n+1} ist Nachfolger aller Aufgaben a_i ($1 \leq i \leq n$).



$$\tau \equiv 1, \quad \tau_i \equiv 0, \quad \tau_e: \text{trivial (z.B. } n+1), m=3$$

Bild III.7 : NP-vollständiges Zuordnungsproblem

Jedem a_i ($1 \leq i \leq n+1$) müssen höchstens $(n+1)$ Boole Variable η_{ij} ($1 \leq j \leq n+1$) zugeordnet werden, da nach spätestens $(n+1)$ Zeitintervallen alle Arbeiten bereits durch einen Prozessor erledigt werden können. Zu minimieren ist dann die Zielfunktion

$$\sum_{j=1}^{n+1} j \eta_{n+1,j} = \min$$

unter den Nebenbedingungen

(i) $\sum_{i=1}^{n+1} \eta_{ij} \leq 3$ (Anzahl der vorhandenen Prozessoren)

(ii) $\sum_{j=1}^{n+1} \eta_{ij} = 1$ (Einheitsbearbeitungszeit)

(iii) $j(\eta_{ij} - \eta_{n+1,j}) < 0$ ($a_i < a_{n+1}$)

(iv) $\sum_{i=1}^n \mu(a_i) \eta_{ij} \leq 1$ (Betriebsmittelbeschränkung)

Dieses PB-G/U-System besteht aus $3n+2$ linearen Ungleichungen und $n+1$ linearen Gleichungen über insgesamt $(n+1)^2$ Boole Variablen, die ganz erheblich reduziert werden können. Wegen der Vorgänger-

beziehung weiß man

$$\begin{array}{c} \wedge \\ 1 \leq i \leq n \end{array} \quad \eta_{i,n+1} = 0 \wedge \eta_{n+1,1} = 0$$

Deshalb können sofort $n+1$ Boole Variablen aus dem System eliminiert werden. Das gilt auch, wenn man eine schärfere Abschätzung über die Gesamtlänge des Ablaufplans erreicht. Der schlechtest mögliche Fall der sequentiellen Bearbeitung kann nämlich nur dann auftreten, wenn die Betriebsmittelforderungen je zweier Arbeiten $a_i, a_j \in T$ ($1 \leq i \neq j \leq n$) das gesamt verfügbare überschreiten. In allen anderen Fällen ist Parallelarbeit möglich. Eine beliebiger Listenschedule könnte somit eine Abschätzung über die Gesamtlänge liefern, die es erlaubt, weitere Boole-Variablen aus dem PB-G/U-System zu streichen. In diesem Fall könnte das den Faktor $2/3$ ausmachen.

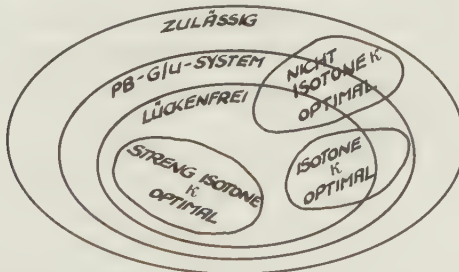


Bild III.8: Hierarchie der Ablaufpläne von Bearbeitungssystemen mit natürlichen Bearbeitungs- und Freigabezeiten bei streng isotonen und nicht isotonen Kostenfunktionen

Allgemein ist ein effektiveres Vorgehen der Lösungsalgorithmen zu erwarten, wenn viele Nebenbedingungen über wenigen Boole-Variablen formuliert sind. Deshalb sollten bereits bei der Erstellung des PB-G/U-Systems alle Möglichkeiten zur Variablenreduktion genutzt werden, die auch zahlreich vorhanden sind, wie obiges Beispiel zeigt. Zusätzliche Vereinfachungen ergeben sich bei Vorhandensein von nicht trivialen Freigabe- und Beendigungszeitpunkten. Jede Arbeit a_i ist nämlich entweder selbst Start-

arbeit oder liegt auf einem Pfad, der von einer solchen kommt. Die Länge eines Pfades ist die Summe der minimalen Bearbeitungszeiten aller darauf liegender Arbeiten. Bei p-inhomogenen Prozessorsystemen kann von der Bearbeitungszeit jeder Aufgabe gesprochen werden. Nimmt man für ein $a_i \in T$ alle solchen Startpfade her, so ist die Länge des längsten trivialerweise eine Abschätzung über die frühest mögliche Fertigstellung von a_i , da Pfade wegen der Vorgängerrelation sequentiell abgearbeitet werden müssen. Setzt man Einheitsbearbeitungszeiten voraus und ist tiefe (a_i) diese Länge, so ist klar, daß alle Boole Variablen η_{ij} mit $1 \leq j < \text{tiefe}(a_i)$ verschwinden müssen, d.h.

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq i \leq n \quad 1 \leq j < \text{tiefe}(a_i) \end{array} \quad \eta_{ij} = 0$$

Analog hierzu kann für jede Aufgabe a_i der längste Pfad zu einer Endarbeit bestimmt werden, die zum Zeitpunkt $T_e = \max \{\tau_e(a_i) : 1 \leq i \leq n\}$ spätestens beendet sein muß. Hat dieser die Länge höhe (a_i), so kann die Endarbeit nur dann zeitgerecht beendet werden, wenn die Boole Variablen η_{ij} mit $T_e - \text{höhe}(a_i) + 1 \leq j \leq T_e$ verschwinden, d.h.

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq i \leq n \quad T_e - \text{höhe}(a_i) + 1 \leq j \leq T_e \end{array} \quad \eta_{ij} = 0$$

Beide Verfahren können dadurch verschärft werden, daß zusätzlich im ersten Fall die Freigabezeitpunkte der Vorgänger von a_i und im zweiten die spätesten Beendigungszeiten $\tau_e(a_j)$ aller Nachfolger a_j in die Überlegungen einbezogen werden. Jede Arbeit a kann frühestens zum Zeitpunkt

$$\tau_f(b) + \min \{\tau_i(b) : 1 \leq i \leq n\}$$

all ihrer direkten Vorgänger $b \in T$ begonnen werden und muß spätestens zum Zeitpunkt

$$\tau_e(c) = \min \{ \tau_i(c) : 1 \leq i \leq n \}$$

ihrer direkten Nachfolger c beendet sein. Im anderen Fall existiert kein zulässiger Ablaufplan. Deswegen liefern diese Werte, iterativ auf den Pfaden ermittelt, eine zusätzliche Abschätzung für das Verschwinden von redundanten Boole Variablen η_{ij} und verschärfen u.U. das vorgeschriebene Bearbeitungsintervall $[\tau_f(a_i), \tau_e(a_i))$. Das Verfahren ist bei einer bestimmten Anwendung so übersichtlich, daß auf eine genaue Formulierung verzichtet werden kann. Im Beispiel 4 (Bild V.5) wurde es angewendet und erzielte dort eine Reduktion von 147 auf 38 Boole Variable.

Darüber hinausgehende Verschärfungen verwenden Verfahren, die bestimmte Strukturen von Untergraphen des Arbeitensystems auf einen Knoten abbilden und dadurch die Anzahl der η_{ij} gering halten. Das wird im ersten Abschnitt des nächsten Kapitels hergeleitet und führt, wie wir sehen werden, zu nicht linearen Pseudo-Boole Nebenbedingungen.

KAPITEL IV: ERWEITERUNGEN DER PROBLEMLÖSUNGEN $(G, B, ex.\underline{\sigma}?)$ UND (G, B, opt_{κ})

In diesem Kapitel werden Erweiterungen der bisher behandelten Zuordnungsprobleme abgeleitet. Der Reihe nach werden vier Themenkreise behandelt: Reduktion von Boole Variablen beim Auftreten "linearer Ketten" im Arbeitengraph, Einbeziehung von natürlichen Bearbeitungszeiten (bisher: Einheitsbearbeitungszeiten) in die Fragestellungen $(G, B, ex.\underline{\sigma}?)$ und (G, B, opt_{κ})

und Berücksichtigung von Verdrängungskosten bei eingeschränkter unterbrechender Verarbeitung. Zwischendurch wird eine Erweiterung des Begriffs p-inhomogenes Prozessorsystem definiert, die es auch erlaubt, Ablaufpläne für uniforme und heterogene Prozessorsysteme [Ecker 77] mit Pseudo-Boole Methoden zu bestimmen.

IV.1 Reduktion von Boole Variablen beim Auftreten linearer Ketten in G

Def. 0 (Kap. IV.1): Es sei $G = (T, <)$ der Arbeitengraph eines p-inhomogenen Arbeitensystems (G, B) . Ein Untergraph $G' = (T', <')$ heißt lineare Kette in G, falls

$$(i) \quad \begin{array}{c} \diagup \quad \diagdown \\ T' \subseteq T^{(i)} \\ 1 \leq i \leq p \end{array}$$

$$(ii) \quad T' = \{a_{i_j} : 1 \leq j \leq l\} \text{ läßt sich in einer aufsteigenden Folge } a_{i_1} < a_{i_2} < a_{i_3} < \dots < a_{i_l} \text{ anordnen}$$

$$(iii) \quad \begin{array}{c} \diagup \quad \diagdown \\ a \in T \setminus T' \quad 2 \leq j \leq l \end{array} (a, a_{i_j}) \notin < \wedge \quad \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq j \leq l-1 \end{array} (a_{i_j}, a) \notin <$$

$$(iv) \quad \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq j_1, j_2 \leq l \end{array} \mu(a_{i_{j_1}}) = \mu(a_{i_{j_2}})$$

$$(v) \quad \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq j_1, j_2 \leq l \end{array} \tau_f(a_{i_{j_1}}) = \tau_f(a_{i_{j_2}}) \wedge \tau_e(a_{i_{j_1}}) = \tau_e(a_{i_{j_2}})$$

Die Definition besagt, daß alle Aufgaben einer linearen Kette vom gleichen Prozessortyp bearbeitbar sein müssen (Bed. (i)), daß ihre Betriebsmittelforderungen identisch sind (Bed. (iv)) und ebenso die Freigabe- und Beendigungszeitpunkte (Bed. v)). Die letzte Eigenschaft kann abgeschwächt werden. Es genügt z.B. zu fordern, daß

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 < j \leq 1 \end{array} \quad \tau_f(a_{i_j}) \leq \tau_f(a_{i_1}) + \sum_{k=1}^{j-1} \tau_{m_i}(a_{i_k}) \wedge$$

(v)*

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq j < 1 \end{array} \quad \tau_e(a_{i_j}) \geq \tau_e(a_{i_1}) - \sum_{k=j+1}^1 \tau_{m_i}(a_{i_k})$$

In Verbindung mit (ii) und (iii) ist das für die Fragestellungen $(G, B, \text{ex. } \underline{\sigma}?)$ und $(G, B, \text{opt}_{\kappa})$ äquivalent mit (v), weil wegen der Vorgängerbeziehung

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq j \leq 1 \end{array} \quad \tau_f(a_{i_j}) = \tau_f(a_{i_1}) \wedge \tau_e(a_{i_j}) = \tau_e(a_{i_1})$$

angenommen werden kann. Die Eigenschaft (iii) ist bedeutsam, weil darin gefordert wird, daß nur die Anfangsarbeit der linearen Kette direkter Nachfolger eine Aufgabe außerhalb der T' sein darf und nur die Endarbeit a_{i_1} die Abarbeitung des Restgraphen im Sinn der Vorgängerrelation beeinflussen kann.

Ein solcher Untergraph kann durch einen einzigen Knoten a' ersetzt werden, der auf dem Prozessortyp i die Bearbeitungszeit

$$\sum_{j=1}^1 \tau_{m_i}(a_{i_j})$$

benötigt, ansonsten die gleichen Betriebsmittelanforderungen, Freigabe- und Beendigungszeitpunkte wie jede Arbeit der linearen Kette hat. Die Einbettung von a' in den Restgraphen geschieht so, daß jede Kante $(a, a_{i_1}) \in <$ eines Knotens a außerhalb der linearen Kette durch eine Kante $(a, a') \in <^*$ im modifizierten Graphen $G^* = (T^*, <^*)$ ersetzt wird und ebenfalls für jede Vorgängerbeziehung $(a_{i_1}, a) \in <$ eine Kante $(a', a) \in <^*$ im neuen Graphen G^* gezogen wird. Durch diese Konstruktion ist

gewährleistet, daß a' genau dann bearbeitet werden kann, wenn im alten Arbeitengraph G a_{i_1} startbar ist und jeder Nachfolger von a' in G^* genau dann, wenn a_{i_1} in G beendet wurde. Um die Äquivalenz von (G, B) und (G^*, B^*) sicherzustellen, müssen Unterbrechungen bei der Bearbeitung von a' zu den Zeitpunkten $\tau_{m_i}(a_{i_1})$, $\tau_{m_i}(a_{i_1}) + \tau_{m_i}(a_{i_2})$, ...,

$l-1$
 $\sum_{j=1} \tau_{m_i}(a_{i_j})$ nach dem Start von a' zugelassen werden. Für Einheitsbearbeitungszeiten $\tau_{m_i}(a_{i_j}) = 1$, wie sie im Kap. III allgemein vorausgesetzt waren, läßt sich das einfach formulieren.

Unter den dortigen Annahmen konnte in den Lemma 7 (Kap. III.2) und 10 (Kap. III.3) gezeigt werden, daß es sowohl bei der Fragestellung $(G, B, ex.\underline{\sigma}?)$ als auch bei (G, B, opt_x) genügt, lückenfreie Ablaufpläne zu betrachten, die dann die Eigenschaft haben, Bearbeitungen auf Intervallgrenzen beginnen zu lassen und deshalb durch Boole Variable η_{ij} beschrieben werden können. Die Ersetzung einer linearen Kette der Länge l mit Einheitsbearbeitungszeiten durch eine Aufgabe mit der natürlichen Rechenzeit l ändert an der Beschreibungsmöglichkeit von Ablaufplänen durch Boole Variable nichts. Statt für jedes Glied der linearen Kette T_e Boole Variable zu benötigen, ist das nur noch für eine einzige Arbeit a' nötig. Die Ersetzung erspart

$$(l-1) \cdot T_e \text{ Boole Variable.}$$

Eine fortgesetzte Anwendung führt zu einem Arbeitengraph, der schließlich keine linearen Ketten mehr hat und dessen Bearbeitungszeiten im allgemeinen natürliche Zahlen sind. Jeder Aufgabe a_i werden wiederum T_e Boole Variable η_{ij} zugeordnet ($1 \leq j \leq T_e$: späteste Fertigstellung aller Aufgaben $1 \leq i \leq n' \leq n$), die jedoch nicht mehr die Bedingung (ii) von Satz 8 (Kap. III.2)

$$\sum_{j=1}^{T_e} \eta_{ij} = 1 \quad \text{sondern} \quad \sum_{j=1}^{T_e} \eta_{ij} = l_i \in \mathbb{N}$$

erfüllen müssen. Da die Bearbeitung an den Einheitsintervallgrenzen unterbrechend sein darf, können die η_{ij} aufeinanderfolgend die Werte 0 und 1 mehrfach annehmen. Dadurch kann die Vorrangbeziehung nicht mehr durch die Bedingung (iii) von Satz 8 (Kap. III.2) ausgedrückt werden. Es wird dort benützt, daß für alle $1 \leq i \leq n$ genau ein $1 \leq j \leq T_e$ existiert mit

$\eta_{ij} = 1$. Um etwas äquivalentes für den Fall zu bekommen, in dem mehrere η_{ij} den Wert 1 annehmen, versucht man, für jede Aufgabe a_i die Indizes j_1 und j_2 zu bestimmen, die das erste und letzte Vorkommen einer 1 anzeigen. Die Durchführung von a_i beginnt dann im Zeitpunkt $j_1 - 1$ und endet bei $t = j_2$. Zwischen-
durch können Bearbeitungsunterbrechungen auftreten.

Eine Möglichkeit zur Bestimmung dieser Werte liefert folgende Substitution:

Seien η_{ij} ($1 \leq j \leq T$) die Boole Variablen einer Aufgabe $a_i \in T$ mit natürlicher Bearbeitungszeit. Dann erfüllen die Boole Ausdrücke

$$\begin{array}{l} \text{e)} \\ \text{a)} \end{array} \quad \begin{array}{l} z_{ik} = \eta_{ik} \prod_{j=k+1}^{T_e} \bar{\eta}_{ij} \quad \text{und} \\ z_{ik} = \left(\prod_{j=1}^{k-1} \bar{\eta}_{ij} \right) \eta_{ik} \end{array}$$

die Eigenschaften

$$(*) \quad \sum_{k=1}^{T_e} z_{ik}^{(e)} = 1 = \sum_{k=1}^{T_e} z_{ik}^{(a)} \quad \text{und}$$

$$2*) \quad z_{ik}^{(e)} = 1 \Leftrightarrow k = \max \{j: \eta_{ij} = 1 \wedge 1 \leq j \leq T_e\}$$

$$z_{ik}^{(a)} = 1 \Leftrightarrow k = \min \{j: \eta_{ij} = 1 \wedge 1 \leq j \leq T_e\}$$

$$\text{D.h.} \quad \sum_{k=1}^{T_e} k z_{ik}^{(e)} = j_2 \quad \text{und} \quad \sum_{k=1}^{T_e} k z_{ik}^{(a)} = j_1$$

Folglich sind die Sätze 8 (Kap. III.2) und 11 (Kap. III.3) auf eine breitere Klasse von Zuordnungsproblemen erweiterbar und insbesondere geeignet, lineare Ketten aus Arbeitsgraphen zu entfernen unter gleichzeitiger Verschärfung der PB-Nebenbedingungen.

Satz 12 (Kap. IV.1)

Es sei (G, B) ein p -inhomogenes Arbeitensystem mit natürlichen Bearbeitungszeiten und Freigabezeitpunkten. Dann existiert ein zulässiger Ablaufplan, der Unterbrechungen nach Einheitsbearbeitungsintervallen erlaubt, genau dann, wenn es $n \cdot T_e$ (n : Anzahl der Arbeiten; $T_e = \max \{\tau_e(a) : a \in T\}$) Boole Variablen η_{ij} ($1 \leq i \leq n$, $1 \leq j \leq T_e$) gibt, die folgenden Bedingungen genügen:

$$(i) \quad \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq k \leq p \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq j \leq T_e \end{array} \quad \sum_{a_i \in T(k)} \eta_{ij} \leq m_k - m_{k-1}$$

$$(ii) \quad \begin{array}{c} \diagup \quad \diagdown \\ 1 \leq i \leq n \end{array} \quad \sum_{j=1}^{T_e} \eta_{ij} = \tau(a_i) = \sum_{j=\tau_f(a_i)+1}^{\tau_e(a_i)} \eta_{ij} \quad (\tau(a_i) \in \mathbb{N})$$

$$(iii) \quad \begin{array}{c} \diagup \\ a_i, a_j \in T \end{array} \quad (a_i < a_j \Rightarrow \sum_{k=1}^{T_e} k z_{ik}^{(e)} < \sum_{k=1}^{T_e} k z_{jk}^{(a)}), \text{ wobei}$$

$$\begin{array}{c} \diagup \\ 1 \leq k \leq T_e \end{array} \quad z_{ik}^{(e)} = \eta_{ik} \left(\prod_{j=k+1}^{T_e} \bar{\eta}_{ij} \right) \wedge z_{ik}^{(a)} = \left(\prod_{j=1}^{k-1} \bar{\eta}_{ij} \right) \eta_{ik}$$

$$(iv) \quad \begin{array}{c} \diagup \\ 1 \leq i \leq s \end{array} \quad \begin{array}{c} \diagup \\ 1 \leq j \leq T_e \end{array} \quad \sum_{i=1}^n \mu_1(a_i) \eta_{ij} \leq 1.$$

Bew.:

Er ergibt sich analog zum Beweis von Satz 8 (Kap. III.2) mit Hilfe der oben genannten Aussagen (*) und (2*) über die neuen Boole Variablen $z_{ik}^{(e)}$ und $z_{ik}^{(a)}$ ($1 \leq k \leq T_e$). $\square$

Die Lösung solcher Fragestellungen führt zu einem nicht linearen PB-G/U-System. Dafür sorgen die Substitutionen $z_{ik}^{(a)}$ und $z_{ik}^{(e)}$. Ein Analogon zu Satz 11 (Kap. III.3) kann mit den gleichen Schlüssen formuliert werden.

Satz 13 (Kap. IV.1)

Sei (G, B) ein p -inhomogenes Arbeitssystem mit natürlichen Bearbeitungszeiten und Freigabezeitpunkten. Dann ist bei einer isotonen Kostenfunktion κ die Bestimmung eines κ -minimalen Ablaufplans, der Unterbrechungen nach Einheitsbearbeitungsintervallen erlaubt, äquivalent mit dem Lösen folgender Pseudo-Boole Optimierungsaufgabe über den Variablen η_{ij} ($1 \leq i \leq n$: Anzahl der Arbeiten; $1 \leq j \leq T_e$: spätester Beendigungszeitpunkt aller Arbeiten) mit der

$$\text{Zielfunktion: } \kappa \left(\sum_{j=1}^{T_e} j z_{1j}^{(e)}, \sum_{j=1}^{T_e} j z_{2j}^{(e)}, \dots, \sum_{j=1}^{T_e} j z_{nj}^{(e)} \right) = \min$$

und Nebenbedingungen wie in Satz 12 (Kap. IV.1).

Bew.:

Die Argumentation ist die gleiche wie im Satz 11 (Kap. III.3) unter Benützen der Beziehungen (*) und (2*). □

IV.2 Nichtunterbrechende Bearbeitung bei natürlichen Rechenzeiten

Unterbrechungen auf Einheitsintervallgrenzen können bei der Bearbeitung einer Aufgabe a_i ($1 \leq i \leq n$) mit natürlicher Rechenzeit $\tau(a_i)$ unterdrückt werden, wenn es gelingt, zusätzliche Nebenbedingungen über den Boole Variablen η_{ij} ($1 \leq j \leq T_e$) zu formulieren, die es verhindern, daß zwischen zwei beliebigen $\eta_{ij_1} = 1$ und $\eta_{ij_2} = 1$ ($1 \leq j_1 < j_2 \leq T_e$) andere η_{ij} ($j_1 < j < j_2$) den Wert 0 annehmen können. Das bedeutet nichts anderes, als daß in der Folge der η_{ij} ($1 \leq j \leq T_e$) einer Arbeit $a_i \in T$ kein η_{ij} Null sein darf, wenn η_{ij-1} und η_{ij+1} den Wert eins annehmen, keine zwei η_{ij} und η_{ij+1} aufeinanderfolgend verschwinden dürfen, wenn deren Nachbarn η_{ij-1} und η_{ij+2} eins sind, usw.. Mit anderen Worten, es dürfen keine solchen "1-er Lücken", "2-er Lücken", ... "k-Lücken" ... und schließlich " $(T_e - \tau(a_i))$ -Lücken" von verschwindenden Boole Variablen auftreten, wenn die angrenzenden Nachbarn den Wert eins annehmen. Eine Lücke der Länge k zwischen den Variablen η_{ij} und η_{ij+k+1} tritt offensichtlich genau dann auf, wenn

$$\eta_{ij} \left(\prod_{l=j+1}^{j+k} \overline{\eta_{il}} \right) \eta_{ij+k+1} = 1.$$

Derartige k-Lücken dürfen an keiner Stelle auftreten, d.h. obige Beziehung muß für alle $1 \leq j \leq T_e - k - 1$ gelten. Da überhaupt nur Lücken der maximalen Länge $T_e - \tau(a_1)$ auftreten können, sichert die Nebenbedingung

$$(v^*) \quad \sum_{k=1}^{T_e - \tau(a_1)} \sum_{j=1}^{T_e - k - 1} (\eta_{ij} \cdot \eta_{i,j+k+1}) = 0$$

eine nichtunterbrechende Aufgabenbearbeitung von a_1 .

Die Bedingungen in den Sätzen 12 und 13 (Kap. IV.1) müssen um diese Gleichheiten erweitert werden, um die Existenzfrage und das Auffinden eines optimalen Ablaufplans unter den dortigen Voraussetzungen für eine nichtunterbrechende Bearbeitung ebenfalls beantworten zu können. Rationale Rechenzeiten sind dann nur noch eine triviale Fortführung der natürlichen.

IV.3 Verallgemeinerung eines p-inhomogenen Prozessorsystems

In den bisherigen Abschnitten konnten Pseudo-Boole Methoden zur Lösung von Zuordnungsproblemen in p-inhomogenen Prozessorsystemen verwendet werden. Diese schließen homogene Prozessorsysteme ein. Jede Arbeit kann zwar auf mehreren Prozessoren gerechnet werden, aber auf all diesen mit gleicher Rechenzeit. In der Praxis ist auch der Fall interessant, in dem eine Aufgabe von unterschiedlichen Prozessoren mehr oder weniger schnell erledigt werden kann. Insbesondere sind Systeme denkbar, in denen die Leistung zweier Prozessoren global durch einen Faktor beschrieben werden kann, sog. uniforme Prozessorsysteme [Ecker 77].

Wir gehen noch einen Schritt weiter und zeigen, daß Pseudo-Boole Methoden sogar im allgemeinsten Fall eines heterogenen Prozessorsystems die Zuordnungsprobleme $(G, B, ex, \underline{g}?)$ und $(G, B, opt, \underline{n})$ lösen können, falls die Bearbeitungszeiten und Freigabezeitpunkte natürlich sind.

Def. P(Kap. IV.3): Ein Arbeitensystem (G, B) mit m Prozessoren und n Aufgaben besitzt ein (p_1, p_2) -heterogenes Prozessorsystem, wenn eine Folge natürlicher Zahlen m_i ($1 \leq i \leq p_1 \in \mathbb{N}$) existiert

$$0 = m_0 < m_1 < m_2 < \dots < m_{p_1} = m \text{ (Anzahl der Prozessoren)}$$

und eine Partition der Arbeitenmenge

$$T = \bigcup_{i=1}^{p_2} T^{(i)}$$

mit den Eigenschaften

$$1 \leq i \leq p_2 \quad I_i \subseteq \{1, 2, \dots, p_1\} \quad a \in T^{(i)} \quad k \in I_i$$

$$(m_{k-1} < j_1, j_2 \leq m_k \Rightarrow \tau_{j_1}(a) = \tau_{j_2}(a) \neq \infty)$$

$$\wedge \quad k \notin I_i \quad (m_{k-1} < j < m_k \Rightarrow \tau_j(a) = \infty).$$

Die Definition besagt, daß in (p_1, p_2) -heterogenen Prozessorsystemen Arbeiten auch auf mehreren Prozessortypen unterschiedlich schnell durchgeführt werden können. Gehört a zu $T^{(i)}$ ($1 \leq i \leq p_2$), dann beschreibt die Indexmenge I_i alle in Frage kommenden Typen. Der Rest des Prozessorsystems kann a nicht bearbeiten.

Ein (p,p) -heterogenes Prozessorsystem mit $I_1 = \{i\}$ ($1 \leq i \leq p$) ist p -inhomogen, so daß tatsächlich eine Verallgemeinerung vorgenommen wurde. Diese umschließt auch heterogene und uniforme Prozessorsysteme (gemäß Def. C (Kap. II.2)), die (m,n) -heterogen im Sinn obiger Definition sind. Die Folge der $(m_i)_{1 \leq i \leq m}$ ist identisch mit $(i)_{1 \leq i \leq m}$ und die Aufgabenpartition ist die kleinstmögliche:

$$T^{(i)} = \{a_i\} \quad (1 \leq i \leq n).$$

Die Indexmenge I_1 beschreibt die Einzelprozessoren, die $a_i \in T$ überhaupt bearbeiten können. Für zulässige Ablaufpläne $\underline{\sigma}$ von (p_1, p_2) -heterogenen Arbeitssystemen lassen sich folgende Mengen definieren:

$$\bigwedge_{1 \leq k \leq p_1} \hat{T}_k = \{a_i \in T: m_{k-1} < \sigma_2(a_i) \leq m_k\}.$$

Sie beinhalten alle Aufgaben, die jeder Prozessortyp k bei Vorgehen gemäß $\underline{\sigma}$ erledigt und stellen unter der Voraussetzung

$$\bigwedge_{1 \leq k \leq p_1} \hat{T}_k \neq \emptyset$$

eine Partition von T dar. Sollte tatsächlich

ein $\hat{T}_k$ leer sein, dann war der Prozessortyp k für die Bearbeitung von $G=(T, <)$ überflüssig, so daß die Gesamtzahl der vorhandenen Prozessoren um den Anteil $(m_k - m_{k-1})$ gekürzt werden kann. Nehmen wir also o.B.d.A. an, daß

$$\bigwedge_{1 \leq k \leq p_1} \hat{T}_k \neq \emptyset$$

Dann ist $\underline{\sigma}$ auch zulässiger Ablaufplan für das folgende Arbeitssystem (G', B') :

$G' = G, \underline{\mu}' = \mu, \tau_f' = \tau_f, \tau_e' = \tau_e, \kappa' = \kappa$ aber:

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq k \leq m \end{array} \quad \begin{array}{c} \diagup \quad \diagdown \\ a_i \in \hat{T}_k \end{array} \quad \tau_j'(a_i) = \begin{cases} \tau_j(a_i), & \text{falls } m_{k-1} < j \leq m_k \\ \infty & \text{sonst.} \end{cases}$$

Dieses Arbeitensystem ist p_1 -inhomogen, die zugehörige Arbeitspartition stimmt mit $\bigcup_{k=1}^{p_1} \hat{T}_k$ überein. (G', B') ist durch eine Beschränkung in den Rechenzeiten dem ursprünglich gegebenem (p_1, p_2) -heterogenen Prozessorsystem zugeordnet worden. Die Existenz von zulässigen Ablaufplänen für (G, B) und (G', B') steht also in einem gewissen Zusammenhang, der nachfolgend präzisiert wird und es schließlich erlaubt, Pseudo-Boole Methoden, die p -inhomogene Fragestellungen lösen, auf (p_1, p_2) -heterogene Prozessorsysteme zu übertragen.

Def. Q (Kap. IV.3): Ein Arbeitensystem (G', B') heißt dem (p_1, p_2) -heterogenen Arbeitensystem (G, B) adjungiert, wenn es sich nur in den Rechenzeiten $\tau' : T \rightarrow (\mathbb{R}^*)^m$ von (G, B) unterscheidet, und zwar in folgender Weise: sind I_i alle Prozessortypen, die $a_i \in T$ bearbeiten können, dann existiert genau ein Prozessortyp $k_i \in I_i$ vom dem a_i im Bearbeitungssystem B' gerechnet werden kann.

Ein adjungiertes Arbeitensystem ist stets p_1 -inhomogen. Aufgrund obiger Bemerkung weiß man, daß die Existenz eines zulässigen Ablaufplans für ein adjungiertes Arbeitensystem (G', B') eine notwendige Bedingung für die Existenz eines zulässigen Ablaufs des gegebenen (p_1, p_2) -heterogenen Arbeitensystems darstellt. Die Umkehrung gilt trivialerweise, so daß folgendes Lemma formuliert werden kann.

Lemma 14 (Kap. IV.3)

Ein zulässiger Ablaufplan des (p_1, p_2) -heterogenen Arbeitensystems (G, B) existiert genau dann, wenn ein adjungiertes Arbeitensystem (G', B') einen zulässigen Ablauf kennt.

Bew.:

Er ergibt sich aus den Vorbemerkungen. $\square$

Da adjungierte Arbeitensysteme p_1 -inhomogen sind, können sie mit den Pseudo-Boole Methoden der früheren Kapitel und Abschnitte bzgl. der Fragestellungen $(G, B, \text{ex. } \underline{\sigma}?)$ und $(G, B, \text{opt. } \kappa)$ unter den dortigen Voraussetzungen gelöst werden. Wegen obigem Lemma 14 liegt die Vermutung nahe, diese Algorithmen effektiv auf den (p_1, p_2) -heterogenen Fall übertragen zu können. In jedem Fall könnten mit den früheren Methoden alle adjungierten Arbeitensysteme einzeln untersucht (es sind nur endliche viele) und hieraus eine Antwort für das (p_1, p_2) -heterogene Prozessorsystem gefunden werden. Hier soll ein besseres Verfahren Anwendung finden.

In den Lemmas 7 (Kap. III.2) und 10 (Kap. III.3) konnte unabhängig von speziellen Annahmen über das Prozessorsystem - lediglich die Kostenfunktion mußte isoton sein - gezeigt werden, daß es genügt, lückenfreie zulässige Ablaufpläne zu betrachten, deren Arbeitsfunktionen η_{ij} ($1 \leq j \leq n$) unter der Annahme natürlicher Bearbeitungszeiten und Freigabezeitpunkte durch Boole Variable η_{ij} ($1 \leq j \leq T_e = \max\{\tau_e(a) : a \in T\}$) diskretisierbar sind. Wird eine Arbeit mit verschiedenen Rechenzeiten auf unterschiedlichen Prozessortypen bedient, so empfiehlt sich die typenspezifische Erweiterung von η_{ij} ($1 \leq j \leq T_e$).

Zu diesem Zweck werden für alle $k \in I_1$ (alle Typen, die $a_i \in T$ überhaupt bearbeiten zu können) Boole Variable η_{ij}^k eingeführt und zusätzlich eine Steuergröße η_1^k , die den Wert 1

genau dann annimmt, wenn a_i gemäß einem zulässigen Ablaufplan $\underline{g}$ auf dem Prozessortyp k abgewickelt wird. Die Aufgabe $a_i \in T$ liegt in einer einzigen Menge $T^{(1_i)}$ der Partition $(T^{(i)})_{1 \leq i \leq p_2}$ von T und kann deswegen von allen Prozessortypen gerechnet werden, die in der Indexmenge I_{1_i} zusammengefaßt sind. Ohne Einschränkung sei in allen Fällen die Rechenzeit nicht länger als das Bearbeitungsintervall der Aufgabe a_i

$$\bigwedge_{k \in I_{1_i}} \tau_{mk}(a_i) \leq \tau_e(a_i) - \tau_f(a_i)$$

Ansonsten erzwingt dies eine Umwandlung des (p_1, p_2) -heterogenen Prozessorsystems in ein (m, n) -heterogenes, in dem für jede Aufgabe $a_i \in T$ die zur Berechnung brauchbaren Prozessortypen einzeln beschrieben sind.

Mit diesen Überlegungen lassen sich analoge Aussagen zu den früheren Sätzen über (p_1, p_2) -heterogene Prozessorsysteme formulieren, die folgende Begriffe verwenden:

l_i : ($1 \leq l_i \leq p_2$) Index der Partitionsmenge, in der eine Aufgabe $a_i \in T$ ($1 \leq i \leq n$) liegt;

I_{1_i} : Indexteilmenge von $\{1, 2, \dots, p_1\}$, die beschreibt, welche Prozessortypen eine Aufgabe $a_i \in T$ ($1 \leq i \leq n$) ausführen können;

$\eta_{1_i}^k$: ($k \in I_{1_i}$) $\eta_{1_i}^k = 1 \Leftrightarrow$ die Aufgabe $a_i \in T$ ($1 \leq i \leq n$) wird gemäß einem zulässigen Ablaufplan $\underline{g}$ vom Prozessortyp k bearbeitet;

$$\eta_{ij}^k : (k \in I_{1_i}, 1 \leq j \leq T_e = \max \{\tau_e(a) : a \in T\}) \quad \eta_{ij}^k \cdot \eta_{i1}^k = 1 \Leftrightarrow$$

die Aufgabe $a_i \in T$ ($1 \leq i \leq n$) wird im Intervall $[j-1, j]$ vom Prozessortyp k bearbeitet;

T_k : ($1 \leq k \leq p_1$) $T_k = \{a_i \in T : k \in I_{1_i}\}$ ist die Menge aller Arbeiten a_i , die vom Prozessortyp k gerechnet werden können.

Satz 15 (Kap. IV.3)

Sei (G, B) ein (p_1, p_2) - heterogenes Arbeitensystem mit natürlichen Bearbeitungs- und Freigabezeitpunkten. Dann existiert ein zulässiger, nichtunterbrechender Ablaufplan $\underline{a}$ genau dann, wenn es für jede Aufgabe $a_i \in T$ Boole Variable η_{i1}^k und η_{ij}^k ($k \in I_{1_i}$, $1 \leq j \leq T_e = \max \{\tau_e(a) : a \in T\}$) gibt, die folgenden Bedingungen genügen:

$$(0) \quad \bigwedge_{1 \leq i \leq n} \sum_{k \in I_{1_i}} \eta_{i1}^k = 1$$

$$(i) \quad \bigwedge_{1 \leq k \leq p_1} \bigwedge_{1 \leq j \leq T_e} \sum_{a_i \in T_k} \eta_{ij}^k \cdot \eta_{i1}^k \leq m_k - m_{k-1}$$

$$(ii) \quad \bigwedge_{1 \leq i \leq m} \bigwedge_{k \in I_{1_i}} \sum_{j=1}^{T_e} \eta_{ij}^k = \tau_m(a_i) = \sum_{j=\tau_f(a_i)+1}^{\tau_e(a_i)} \eta_{ij}^k$$

$$(iii) \quad \bigwedge_{a_i, a_j \in T} (a_i < a_j \Rightarrow$$

$$\sum_{l=1}^{T_e} 1 \left(\sum_{k \in I_{1_l}} (e_{l1}^k \cdot \eta_{i1}^k) \right) < \sum_{j=1}^{T_e} 1 \left(\sum_{k \in I_{1_j}} (a_{j1}^k \cdot \eta_{j1}^k) \right),$$

wobei gilt:

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq i \leq n \quad 1 \leq j \leq T_e \end{array} e_{ij}^k = \eta_{ij}^k \prod_{l=j+1}^{T_e} \bar{\eta}_{il}^k \quad \text{und}$$

$$a_{ij}^k = \left(\prod_{l=1}^{j-1} \bar{\eta}_{il}^k \right) \eta_{ij}^k$$

(iv)

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq l \leq s \quad 1 \leq j \leq T_e \end{array} \prod_{i=1}^n \mu_l(a_i) \left(\sum_{k \in I_{1i}} \eta_{ij}^k \eta_i^k \right) \leq 1$$

(v)

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq i \leq n \quad l \in I_{1i} \end{array}$$

$$\sum_{k=1}^{T_e - \tau_{m_1}(a_i)} \sum_{j=1}^{T_e - k - 1} \eta_{ij}^1 \left(\prod_{m=j+1}^{j+k} \bar{\eta}_{im}^1 \right) \eta_{ij+k+1}^1 = 0$$

Bew.:

1. Schritt:

Sei $\underline{\sigma}$ ein zulässiger Ablaufplan des (p_1, p_2) -heterogenen Arbeitensystems (G, B) und damit nach Lemma 14 (Kap. IV.2) auch zulässig für ein adjungiertes p_1 -inhomogenes Arbeitensystem (G', B') mit

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq k \leq p_1 \quad a_i \in \hat{T}_k \end{array} \tau_j'(a_i) = \begin{cases} \tau_j(a_i), & \text{falls } m_{k-1} < j \leq m_k \\ \infty & \text{sonst.} \end{cases}$$

$$\text{wobei } \hat{T}_k = \{a_i \in T: m_{k-1} < \sigma_2(a_i) \leq m_k\}$$

Nach Satz 12 (Kap. IV.1) existieren somit Boole Variable η_{ij} ($1 \leq i \leq n$, $1 \leq j \leq T_e$), die den dortigen Bedingungen (i) - (iv) genügen. Sie können zu η_{ij}^k und $\bar{\eta}_{ij}^k \in \{0, 1\}$ ($1 \leq i \leq n$, $1 \leq j \leq T_e$, $k \in I_{1i}$) erweitert werden und lösen mit folgender Festlegung das PB-G/U-System (0) - (v) des Satzes:

$$(*) \quad \bigwedge_{1 \leq k \leq p_1} \bigwedge_{a_i \in \hat{T}_k} \eta_{ij}^k = 1 \wedge \bigwedge_{1 \in I_{1_i} \setminus \{k\}} \eta_{ij}^1 = 0$$

$$(2*) \quad \bigwedge_{a_i \in \hat{T}_k} \bigwedge_{1 \leq j \leq T_e} \eta_{ij}^k = -\eta_{ij}$$

$$(3*) \quad \bigwedge_{a_i \in \hat{T}_k} \bigwedge_{1 \leq j \leq T_e} \bigwedge_{1 \in I_{1_i} \setminus \{k\}} \eta_{ij}^1 = \begin{cases} 1, & \text{falls } \tau_f(a_i) + 1 \leq j \leq \tau_f(a_i) + \tau_{m_1}(a_i) \\ 0 & \text{sonst.} \end{cases}$$

Die Gleichungen (0) gelten aufgrund der Festlegung (*), da jedes $a_i \in T$ in genau einem $\hat{T}_k$ liegt.

Die Ungleichungen (i) folgen so: seien $1 \leq k \leq p_1$ und $1 \leq j \leq T_e$. Wegen $\hat{T}_k \subseteq T_k$ und der Tatsache, daß für alle $a_i \in T_k \setminus \hat{T}_k$ $\eta_{ij}^k = 0$ gilt, erhält man:

$$\sum_{a_i \in T_k} \eta_{ij}^k \eta_{ij}^k = \sum_{a_i \in \hat{T}_k} \eta_{ij}^k \stackrel{(2*)}{=} \sum_{a_i \in \hat{T}_k} \eta_{ij} \leq m_k - m_{k-1}$$

da die η_{ij} Lösungen des adjungierten Arbeitensystems sind. (ii) folgt mit der gleichen Argumentation aus (2*), falls $k \in I_{1_i}$ dasjenige ist mit $\sigma_2(a_i) = k$. Für die restlichen $1 \in I_{1_i} \setminus \{k\}$ tut es die "Dummy"-Festlegung (3*). Dort sind nämlich genau

τ_{m1} (a) Boole Variable unter den $(\eta_{ij}^1)_{1 \leq j \leq T_e}$ auf 1 festgelegt worden.

Das Einhalten der Ungleichungen (iii) trifft zu, weil

$$\bigwedge_{a_i \in T} \bigwedge_{1 \leq l \leq T_e} \sum_{k \in I_{l_i}} (e_{il}^k \eta_i^k) = e_{il}^{\sigma_2(a_i)}$$

und somit folgende Gleichheit gilt:

$$\bigwedge_{a_i \in T} \sum_{l=1}^{T_e} 1 \left(\sum_{k \in I_{l_i}} e_{il}^k \eta_i^k \right) = \sum_{l=1}^{T_e} 1 e_{il}^{\sigma_2(a_i)}$$

Sind nun $a_i, a_j \in T$ mit $a_i < a_j$, dann ergibt sich (iii) aus der Zulässigkeit von $\underline{g}$ für das adjungierte Arbeitensystem. Die Ungleichungen für das beschränkte Verwenden von teilbaren Betriebsmitteln (Bedingung (iv)) begründen sich analog zu (i)

Über die Gleichheitsbeziehungen $\bigwedge_{1 \leq j \leq T_e} \bigwedge_{1 \leq i \leq n}$

$$\sum_{k \in I_{l_i}} \eta_{ij}^k \cdot \eta_i^k = \eta_{ij}^{\sigma_2(a_i)}, \text{ die (iv) auf die einfacheren}$$

Ungleichungen des adjungierten Arbeitensystems reduzieren und damit deren Gültigkeit beweisen.

(v) ist eine triviale Erweiterung der Bedingung (v)* aus Kap. IV.2, die aufgrund der Festlegung (3*) zulässig ist.

2. Schritt:

Seien η_{ij}^k und η_i^k ($1 \leq i \leq n$; $1 \leq j \leq T_e$, $k \in I_{l_i}$) Lösungen des

PB-G/U-Systems. Wieso existiert dann ein zulässiger Ablaufplan $\underline{g}$ für das (p_1, p_2) -heterogene Prozessorsystem?

Wegen (0) weiß man, daß für alle $a_i \in T$ genau ein $k_i \in I_{1_i}$ existiert mit $\eta_i^{k_i} = 1$. Deshalb können folgende Mengen definiert werden:

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq i \leq p_1 \end{array} \quad T^1(1) = \{a_i \in T : \eta_i^1 = 1\},$$

die somit eine Partition von T bilden. Das erlaubt die Bestimmung eines p_1 -inhomogenen adjungierten Prozessorsystems (G', B') nach der Vorschrift

$$\begin{array}{c} \diagup \quad \diagdown \\ a_i \in T^1(1) \quad 1 \leq j \leq m \end{array} \quad \tau_j'(a_i) = \begin{cases} \tau_j(a), & \text{falls } m_{1-1} < j \leq m_1 \\ \infty & \text{sonst.} \end{cases}$$

Die Variablen $\eta_{ij}^{k_i}$ ($1 \leq i \leq n$, $1 \leq j \leq T_e$) lösen das PB-G/U-System von (G', B') , womit die Existenz eines zulässigen Ablaufplans nach Satz 12 Kap.(IV.1) für das adjungierte Arbeitensystem und wegen Lemma 14 (Kap. IV.3) auch für das ursprüngliche (p_1, p_2) -heterogene folgt. $\square$

Die Bedingung (v) gewährleistet eine nichtunterbrechende Bearbeitung, so daß bei Weglassen dieses Kriteriums Aufgaben nach Einheitsbearbeitungszeiten den Prozessor entzogen bekommen können, obwohl sie noch nicht fertig sind. Die Aussage des letzten Satzes ändert sich dann in der Weise, daß die Frage nach der Existenz von zulässigen Abläufen, die Unterbrechungen auf Intervallgrenzen zulassen, äquivalent mit der Lösbarkeit des PB-G/U-Systems (0) - (iv) ist. Diese Erweiterungsmöglichkeit trifft auch für den nachfolgenden Satz zu, der die konsequente Fortführung des letzten Ergebnisses ist und die Fragestellung $(G, B, \text{opt}_\kappa)$ in (p_1, p_2) -heterogenen Prozessorsystemen behandelt.

Satz 16 (Kap. IV.3)

Sei (G, B) ein (p_1, p_2) -heterogenes Arbeitensystem mit natürlichen Bearbeitungszeiten und Freigabezeitpunkten. Dann ist bei einer isotonen Kostenfunktion κ das Bestimmen eines minimalen zulässigen Ablaufplans [der Unterbrechungen nach Einheitsbearbeitungszeiten zuläßt,] äquivalent mit dem Lösen folgender Pseudo-Boole Optimierungsaufgabe über den Boole Variablen η_{ij}^k und η_i^k ($1 \leq i \leq n$: Anzahl der Arbeiten; $1 \leq j \leq T_e$: spätester Beendigungszeitpunkt aller Arbeiten, $k \in I_{1_i}$: Indexmenge der Prozessortypen, die zur Bearbeitung von a_i in Frage kommen):

$$\text{Zielfunktion; } \kappa \left(\left(\sum_{j=1}^{T_e} \sum_{k \in I_{1_i}} e_{ij}^k \cdot \eta_{ij}^k \right) \mid 1 \leq i \leq n \right) = \min$$

Nebenbedingungen: wie in Satz 15 (Kap. IV.3).

Bew.:

Der Beweis entspricht dem von Satz 15 (Kap. IV.3) unter Berücksichtigung, daß die Beendigung einer Arbeit gemäß dem Ablauf $\underline{a}$, der den Lösungen η_{ij}^k und η_i^k des PB-G/U-Systems der Nebenbedingungen zugeordnet ist, durch den Pseudo-Boole Ausdruck

$$\sum_{j=1}^{T_e} \sum_{k \in I_{1_i}} e_{ij}^k \eta_{ij}^k$$

ermittelt werden kann. $\square$

In beiden vorangehenden Sätzen entstehen hochgradig nicht lineare PB-G/U-Systeme, die z.B. mit der Methode von FORTET [Fortet 60] gelöst werden können. Eine Zusammenfassung der Ergebnisse dieser Arbeit zeigt, daß die Zuordnungsprobleme

(G,B,ex.g?) und (G,B,opt_n) unter sehr allgemeinen Voraussetzungen mit Pseudo-Boole Methoden lösbar sind. Es können definitive Antworten auf diese Fragen gegeben werden bei

beliebigen Vorgängerbeziehungen,
beliebig heterogenen Prozessorsystemen,
beliebigen Forderungen nach teilbaren Betriebsmitteln,
beliebig vorgegebenen Fertigstellungszeitpunkten.

unter den Einschränkungen

rationaler Bearbeitungszeiten,
rationaler frühester Startzeitpunkte
und isotoner Kostenfunktionen.

Die Frage der Komplexität der verwendeten Pseudo-Boole Lösungsalgorithmen ist bisher ungeklärt. Scharfe Abschätzungen sind vermutlich bereits deshalb schwer erzielbar, weil damit NP-vollständige Zuordnungsprobleme gelöst werden können. Die im Rahmen dieser Arbeit gerechneten Probleme zeigen jedoch die Brauchbarkeit solcher Methoden. Von einem Durchprobieren aller Möglichkeiten kann nicht im entferntesten die Rede sein, wie alle auf einer CYBER 172-Rechenanlage gelösten Zuordnungsprobleme zeigen. Sie sind ausführlich im nächsten Kapitel dieser Arbeit beschrieben.

IV.4 Berücksichtigung von Verdrängungskosten

Im Abschnitt IV.2 wurden Bedingungen über die Boole Variablen η_{ij} ($1 \leq i \leq n$; $1 \leq j \leq T_e$) hergeleitet, die das Auftreten von k-Lücken bei der Bearbeitung einer Aufgabe a_i vermeiden. Mit den dort vorkommenden Boole-Ausdrücken ist es auch möglich, k-Lücken zu zählen. Sind η_{ij} ($1 \leq j \leq T_e$) die Boole Variablen der Arbeit a_i , so zählt der Ausdruck

$$\sum_{j=1}^{T_e^{-k-1}} \eta_{ij} \left(\prod_{l=j+1}^{j+k} \bar{\eta}_{il} \right) \eta_{ij+k+1}$$

die Anzahl der Verdrängungen der Länge k bei einer unterbrechenden Bearbeitung von a_i . Er erlaubt weitere Kombinationen, etwa die Angabe aller Verdrängungen.

$$(*) \quad \sum_{k=1}^{T_e^{-\tau(a_i)}} \sum_{j=1}^{T_e^{-k-1}} \eta_{ij} \left(\prod_{l=j+1}^{j+k} \bar{\eta}_{il} \right) \eta_{ij+k+1}$$

oder sogar bei Summation von $(*)$ über alle Arbeiten $a_i \in T$ ($1 \leq i \leq n$) die Anzahl aller Verdrängungen bei der Gesamtbearbeitung von (G, B) . Diese Angaben sind interessant, weil in der Praxis Unterbrechungen bedeutsame Systembelastungen erzeugen und damit zu Kosten führen.

In der Definition der Funktion κ haben wir das bislang nicht berücksichtigt. Die Beendigungszeiten aller Arbeiten haben allein zur Berechnung von Kosten beigetragen. Für nichtunterbrechende Abläufe ist das auch ausreichend. Im Falle zugelassener Verdrängungen bieten sich mehrere Erweiterungen von κ an, z.B. die Variante

$$\kappa = (\mathbb{R}^{\geq 0})^n \times \mathbb{N} \rightarrow \mathbb{R}$$

$$((x_1, x_2, \dots, x_n), v) \mapsto \kappa((x_1, x_2, \dots, x_n), v)$$

Die $(n+1)$. Komponente nimmt die Anzahl aller Verdrängungen auf und kann somit durch

$$v = \sum_{i=1}^n \sum_{k=1}^{T_e^{-\tau(a_i)}} \sum_{j=1}^{T_e^{-k-1}} \eta_{ij} \left(\prod_{l=j+1}^{j+k} \bar{\eta}_{il} \right) \eta_{ij+k+1}$$

ausgedrückt werden. Ein anderer Weg ist ebenfalls denkbar. Die Kostenfunktion kann die Unterbrechungen arbeitenspezifisch berücksichtigen:

$$\kappa : (\mathbb{R}^{\geq 0})^n \times \mathbb{N} \rightarrow \mathbb{R}$$

$$((x_1, x_2, \dots, x_n), (v_1, v_2, \dots, v_n)) \rightsquigarrow \kappa((x_1, x_2, \dots, x_n), (v_1, v_2, \dots, v_n))$$

v_i ($1 \leq i \leq n$) hätte dann die Form

$$v_i = \sum_{k=1}^{T e^{-\tau(a_i)}} \sum_{j=1}^{T e^{-k-1}} \eta_{ij} \left(\prod_{l=j+1}^{j+k} \bar{\eta}_{il} \right) \eta_{ij+k+1}$$

Wie immer man Verdrängungen in die Kostenfunktion eingehen lassen will, sie können stets in die Zielfunktion der Pseudo-Boole Optimierungsaufgaben der Sätze 16 (Kap. IV.3), 13 (Kap. IV.1) und 11 (Kap. III.3) übertragen werden.

Umgekehrt kann es aus praktischen Gründen sinnvoll sein, nur Ablaufpläne zu betrachten, die eine maximale Anzahl von Verdrängungen nicht überschreiten. Durch Anfügen weiterer PB-Nebenbedingungen auf der Grundlage obiger Terme ist dies möglich. Die Ungleichung

$$\sum_{j=1}^{T e^{-4}} \eta_{ij} \left(\prod_{l=j+1}^{j+3} (\bar{\eta}_{il}) \right) \eta_{ij+4} \leq 2$$

läßt z.B. nur Abläufe zu, die bei der Bearbeitung der Aufgabe a_i mit maximal 2 Verdrängungen der Dauer 3 auskommen. Die Anwendungsmöglichkeiten solcher gezielten Einschränkungen sind beinahe unbegrenzt, wenn Zuordnungsprobleme der Form $(G, B, \text{ex. } \underline{0}?)$ und $(G, B, \text{opt. } \kappa)$ mit Pseudo-Boole Methoden gelöst werden.

Das gilt auch, wenn die Anzahl von Prozessoren eines Typs k zeitabhängig variiert. Bisher wurde ja davon ausgegangen, daß für die gesamte Bearbeitung $(m_k - m_{k-1})$ Prozessoren des Typs k zur Verfügung stehen. Diese Globalität ist eigentlich in den bisher bewiesenen Sätzen 8, 11, 12, 13, 15 und 16 nicht nötig, da dort die Bedingung (i) für jeden Zeitpunkt $1 \leq j \leq T_e$ geprüft wird. Statt der globalen Angaben $m_k (1 \leq k \leq p_1)$ könnten auch zeitabhängige Prozessorzahlen $m_k(j) (1 \leq k \leq p; 1 \leq j \leq T_e)$ geführt werden. Derartige dynamische Prozessorsysteme blieben bisher unberücksichtigt, weil Zuordnungsprobleme ohnehin schwierig genug sind. In dieser Arbeit zeigt sich, daß die Prozessordynamik eigentlich gar keine Erweiterung der Zuordnungsproblematik ist, sondern in den äquivalenten Pseudo-Boole Systemen bereits versteckt vorkommt.

KAP. V: PSEUDO-BOOLE METHODEN ZUM LÖSEN DER ZUORDNUNGSPROBLEME ($G, B, \text{ex. } \underline{\sigma} ?$) und (G, B, opt_κ) UND GERECHNETE BEISPIELE

In den Vorkapiteln konnte nachgewiesen werden, daß die Frage nach der Existenz von Ablaufplänen für eine recht allgemeine Klasse von Arbeitensystemen und ebenso das Bestimmen eines κ -optimalen Ablaufs, falls Ablaufpläne überhaupt existieren, unter der schwachen Voraussetzung einer isotonen Kostenfunktion mit Pseudo-Boole Methoden lösbar sind. In diesem Kapitel werden zwei solche Techniken unter hauptsächlich zwei Aspekten vorgestellt. Zum einen zeigt sich, daß die in den Vorkapiteln zur Lösung von Zuordnungsproblemen erzeugten PB-G/U-Systeme teilweise so spezielle Varianten sind, daß die allgemeinen Lösungsalgorithmen verfeinert werden können und somit das Zuordnungsproblem effektiver lösen. Zum anderen soll die vorliegende Arbeit hierdurch eine Geschlossenheit erfahren, die dem Anwender eine Beurteilung der Verfahren hinsichtlich der Fragestellungen ($G, B, \text{ex. } \underline{\sigma} ?$) und (G, B, opt_κ) erlaubt. Das ist vor allem auch deshalb wichtig, weil Komplexitätsmaße solcher

Algorithmen bislang nicht existieren. Es hat sich gezeigt, daß Pseudo-Boole Techniken wenig bekannt sind, obwohl sie viele praktische Probleme lösen. Zahlreiche Beiträge hierzu und eine umfassende Darstellung der Theorie ist von HAMMER (vormals IVANESCU) und RUDEANU gegeben worden [Hammer & Rudeanu 68], die insbesondere Lösungsalgorithmen für lineare und nicht lineare PB-G/U-Systeme und das Minimieren von nicht notwendig linearen PB-Funktionen unter ebensolchen Nebenbedingungen einschließen. Beide Ansätze werden hier knapp hergeleitet, die daraus resultierenden Algorithmen in Richtung auf die Zuordnungsfragen verfeinert und in einer Grobübersicht algorithmisch niedergeschrieben. Bei den Beweisen von bekannten Sätzen wird auf [Hammer & Rudeanu 68] verwiesen.

V.1 Lineare Pseudo-Boole Methoden

Eine Abbildung $f: \{0,1\}^n \rightarrow \mathbb{R}$ ($n \in \mathbb{N}$) heißt Pseudo-Boole Funktion, in Ergänzung zu Boole Funktionen, die als Bildbereiche ebenfalls die Menge $\{0,1\}$ haben. Die Elemente 0 und 1 der Boole Algebra können als reelle Zahlen, die Boole Operationen $\wedge$, $\vee$ und $\neg$ durch Addition und Multiplikation in $\mathbb{R}$ ersetzt werden:

$$\begin{aligned}x_1 \wedge x_2 &= x_1 \cdot x_2 \\x_1 \vee x_2 &= x_1 + x_2 - x_1 x_2 \\\neg x_1 &= \bar{x}_1 = 1 - x_1\end{aligned}$$

D.h. die Boole Algebra ist isomorph in $\mathbb{R}$ einbettbar, die Boole-Funktionen (B-Funktionen) sind eine Variante der Pseudo-Boole-Funktionen (PB-Funktionen). Analog zur Tatsache, daß jede B-Funktion in konjunktiver Normalform darstellbar ist, existiert der sogenannte "Interpolationssatz" für PB-Funktionen. Er besagt, daß jede Abbildung $f: \{0,1\}^n \rightarrow \mathbb{R}$ eindeutig in die Form

$$f(x_1, x_2, \dots, x_n) = \sum_{\alpha_1, \alpha_2, \dots, \alpha_n} c_{\alpha_1, \alpha_2, \dots, \alpha_n} x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$$

gebracht werden kann, wobei die $\alpha_i \in \{0,1\}$ und $x^1=x, x^0=\bar{x}$ gilt. Die Koeffizienten $c_{\alpha_1\alpha_2\ldots\alpha_n}$ sind aus $\mathbb{R}$ und erfüllen

$$c_{\alpha_1\alpha_2\ldots\alpha_n} = f(\alpha_1, \alpha_2, \ldots, \alpha_n)$$

siehe [Hammer & Rudeanu 68] S. 21 ff.

Ist f eine PB-Funktion, dann heit

$$f(x_1, x_2, \ldots, x_n) = r \in \mathbb{R} \quad \text{PB-Gleichung und}$$

$$f(x_1, x_2, \ldots, x_n) <op> r \in \mathbb{R} \quad \text{PB-Ungleichung, falls}$$

$$<op>: <| \leq | > | \geq$$

Die PB-Abbildung f heit linear, wenn folgende Gestalt vorliegt

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^n a_i x_i + b_i \bar{x}_i \quad (a_i, b_i \in \mathbb{R})$$

Jede lineare PB-Funktion kann in die Gestalt

$$f(x_1, x_2, \ldots, x_n) = d + \sum_{i=1}^n a_i x_i^{\alpha_i}$$

gebracht werden, wobei die $a_i \in \mathbb{R}^{\geq 0}$, $d \in \mathbb{R}$ und $\alpha_i \in \{0,1\}$ sind. Um das zu zeigen, gengt folgende Fallunterscheidung:

1. Fall: $a_i > b_i$

$$\Rightarrow a_i x_i + b_i \bar{x}_i = (a_i - b_i) x_i + b_i \quad \text{fr alle } x_i \in \{0,1\}$$

2. Fall: $a_i \leq b_i$

$$\Rightarrow a_i x_i + b_i \bar{x}_i = (b_i - a_i) \bar{x}_i + a_i \quad \text{fr alle } x_i \in \{0,1\}$$

Aus dieser Substitution ergibt sich unmittelbar die allgemeine Form einer linearen PB-Gleichung.

$$(*) \quad \sum_{i=1}^n c_i x_i^{a_i} = d \in \mathbb{R}$$

Ohne Einschränkung kann angenommen werden, daß $c_1 \geq c_2 \geq \dots \geq c_n > 0$ gilt, wozu allenfalls ein Umbenennung oder die Reduktion von Variablen nötig ist. Zur Lösung von (*) sind 8 gegenseitig sich ausschließende Zustände zu betrachten.

Zustand 1: $d < 0$

Dann existiert keine Lösung.

Zustand 2: $d = 0$

Es gibt genau eine Lösung, nämlich:

$$\bigwedge_{1 \leq i \leq n} x_i^{a_i} = 0$$

Zustand 3: $d > 0$ und $c_1 \geq c_2 \geq \dots \geq c_p > d \geq \dots \geq c_n$

Falls eine Lösung existiert, muß gelten:

$$\bigwedge_{1 \leq i \leq p} x_i^{a_i} = 0 \wedge \sum_{i=p+1}^n c_i \cdot x_i^{a_i} = d$$

Zustand 4: $d > 0$ und $c_1 = c_2 = \dots = c_p = d > c_{p+1} \geq \dots \geq c_n$

Es gibt p Lösungen:

$$\bigwedge_{1 \leq i \leq p} (x_i^{a_i} = 1 \wedge \bigwedge_{1 \leq j \neq i \leq p} x_j^{a_j} = 0)$$

Jede andere Lösung erfüllt:

$$\bigwedge_{1 \leq i \leq p} x_i^{a_i} = 0 \wedge \sum_{i=p+1}^n c_i \cdot x_i^{a_i} = d$$

Zustand 5: $d > 0$ und $d > c_i$ für alle $1 \leq i \leq n$ und $\sum_{i=1}^n c_i < d$

Dann existiert keine Lösung.

Zustand 6: $d > 0$ und $d > c_i$ für alle $1 \leq i \leq n$ und $\sum_{i=1}^n c_i = d$

Dann existiert genau eine Lösung:

$$\begin{array}{c} \diagup \quad \diagdown \\ 1 \leq i \leq p \end{array} x_i^{a_i} = 1.$$

Zustand 7: $d > 0$ und $d > c_i$ für alle $1 \leq i \leq n$ und $\sum_{i=1}^n c_i > d$,

$$\text{aber } \sum_{i=2}^n c_i < d$$

Falls eine Lösung existiert, muß für sie gelten:

$$x_i^{a_i} = 1 \text{ und } \sum_{i=2}^n c_i x_i^{a_i} = d - c_1$$

Zustand 8: $d > 0$ und $d > c_i$ für alle $1 \leq i \leq n$ und $\sum_{i=1}^n c_i > d$

$$\text{aber } \sum_{i=2}^n c_i \geq d$$

Falls eine Lösung existiert, muß für sie gelten:

$$\text{entweder } x_i^{a_i} = 1 \text{ und } \sum_{i=2}^n c_i x_i^{a_i} = d - c_i$$

$$\text{oder } x_i^{a_i} = 0 \text{ und } \sum_{i=2}^n c_i x_i^{a_i} = d$$

Beim Lösen einer linearen Pseudo-Boole Gleichung können demnach folgende Zustände auftreten:

- die Gleichung besitzt keine Lösung (Zustände 1 und 5)
- die Gleichung besitzt genau eine Lösung (Zustände 2 und 6)
- die Gleichung wird ersetzt durch eine Gleichung desselben Typs mit jedoch weniger Variablen (Zustände 3, 4 und 7), nachdem u.U. einige Lösungen bereits bestimmt sind (Zustand 4)

- die Gleichung wird ersetzt durch zwei Gleichungen desselben Typs mit jedoch weniger Variablen (Zustand 8)

Durch rekursive Anwendung dieses Verfahrens bekommt man somit die Lösungen. Die 8 Zustände können klassifiziert werden:

Zustände 1,2,3,5,6,7:	die Gleichung (*) determiniert,
Zustand 4	: die Gleichung (*) determiniert partiell,
Zustand 8	: die Gleichung (*) ist undeterminiert.

Jede Determinierung bedeutet entweder, die Lösungsmenge ist endgültig bestimmt (Zustände 1,2,5,6), oder der Wert einer Variablen steht definitiv fest (Zustände 3,7). Im Gegensatz hierzu werden im 4. Zustand p Lösungen bestimmt und die restlichen dadurch auffindbar, daß $n-p$ Variable eine lineare PB-Gleichung bilden, die noch zu lösen ist. Es liegt eine teilweise Determinierung vor. Der Zustand 8 ist der einzig pathologische Fall, weil keine Aussage über eine mögliche Lösung gemacht werden kann, sondern unabhängig voneinander zwei Gleichungen mit jeweils $n-1$ Variablen untersucht werden müssen. Allein in diesem Fall findet ein Durchprobieren aller Werte einer bestimmten Variablen statt, wie dies durch den binären Baum der Tiefe n beschrieben wird, dessen 2^n -Pfade alle in Frage kommenden Wertekombinationen repräsentieren. In den anderen Fällen werden nicht aussichtsreiche Lösungspfade erkannt und "gestrichen", deren Anzahl bei einer partiellen Determinierung des Grades p immerhin 2^{n-p} beträgt. Die Komplexität dieses rekursiven Algorithmus wird also geprägt werden von der Anzahl der auftretenden undeterminierten Zustände 8.

Vollkommen analog zur allgemeinen Form der linearen PB-Gleichung (*) kann eine Ungleichung nach einer u.U. nötigen Multiplikation mit -1 und nach einem Umsortieren in die Gestalt (2*) gebracht werden.

$$(2^*) \quad \sum_{i=1}^n c_i x_i^{a_i} \geq d \in \mathbb{R}$$

Sind die Koeffizienten c_i natürliche Zahlen (oder auch rationale), dann trifft dies auch für strikte Ungleichungen zu. In [Hammer & Rudeanu 68] wird gezeigt, daß diese Unterscheidung auch bei allgemeinen Ungleichungen nicht gemacht werden muß. Hier spielt das ohnehin keine Rolle, da bei Zuordnungsproblemen nur $\leq$ -Beziehungen auftreten (Betriebsmittelforderungen) oder die PB-Funktionen den Wertebereich $\mathbb{N}$ haben. (2^*) kennt 6 sich gegenseitig ausschließende Zustände.

Zustand 1: $d \leq 0$

Jedes Wertetupel löst die Ungleichung.

Zustand 2: $d > 0$ und $c_1 \geq c_2 \geq \dots \geq c_p \geq d > c_{p+1} \geq \dots \geq c_n$

Jedes Wertetupel mit $\bigwedge_{1 \leq i \leq p} x_i^{a_i} = 1$ ist eine

Lösung. Ein anderes Tupel kann nur Lösung sein, wenn

$$\sum_{i=p+1}^n c_i x_i^{a_i} \geq d$$

Zustand 3: $d > 0$ sogar $d > c_i$ für alle $1 \leq i \leq n$ und

$$\sum_{i=1}^n c_i < d$$

Dann ist die Ungleichung unlösbar.

Zustand 4: $d > 0$, sogar $d > c_i$ für alle $1 \leq i \leq n$ und $\sum_{i=1}^n c_i = d$

Dann existiert genau eine Lösung, nämlich:

$$\bigwedge_{1 \leq i \leq n} x_i^{a_i} = 1.$$

Zustand 5: $d > 0$, sogar $d < c_i$ für alle $1 \leq i \leq n$ und $\sum_{i=1}^n c_i > d$,
aber $\sum_{i=2}^n c_i < d$

Falls Lösungen existieren, müssen diese $x_1^{\alpha 1} = 1$ erfüllen.

Zustand 6: $d > 0$, sogar $d > c_i$ für alle $1 \leq i \leq n$ und $\sum_{i=1}^n c_i > d$,
aber $\sum_{i=2}^n c_i \geq d$

Jedes Tupel $(x_1, x_2, \dots, x_n)$ ist Lösung der Ungleichung,

falls entweder $x_1^{\alpha 1} = 1$ und $\sum_{i=2}^n c_i x_i^{\alpha i} \geq d - c_1$

oder $x_1^{\alpha 1} = 0$ und $\sum_{i=2}^n c_i x_i^{\alpha i} \geq d$.

Damit ist ein rekursives Lösen von (2*) ebenfalls möglich, wo-
bei wie im Fall der linearen PB-Gleichung

- keine Lösung existieren kann oder die Ungleichung trivial ist (Zustände 3,1),
- genau ein Wertetupel $(x_1, x_2, \dots, x_n)$ Lösung ist (Zustand 4)
- die Ungleichung durch eine ebensolche mit weniger Variablen ersetzt werden kann (Zustände 2,5), nachdem Lösungen u.U. gefunden wurden (Zustand 2)
- oder stattdessen zwei lineare PB-Ungleichungen mit weniger Variablen zu lösen sind (Zustand 6).

Eine Klassifizierung dieser Zustände sieht sinnvollerweise so aus:

Zustände 1,3,4,5: die Ungleichung (2*) determiniert,
Zustand 2 : die Ungleichung (2*) determiniert partiell,
Zustand 6 : die Ungleichung (2*) ist undeterminiert.

Die rekursive Anwendung dieser Zustandsuntersuchung liefert wie im Fall der linearen PB-Gleichung alle Lösungen, wobei beide Verfahren auf einer "kanonischen" Darstellung der Form (*) bzw. (2*) beruhen und recht einfach auf lineare PB-G/U-Systeme erweitert werden können. Es geht dann darum, Boole Variable so zu bestimmen, daß gleichzeitig mehrere lineare PB-Gleichungen $f_i(x_1, x_2, \dots, x_n) = d_i$ ($1 \leq i \leq m$) und lineare PB-Ungleichungen $f_j(x_1, x_2, \dots, x_n) \langle op \rangle d_j$ ($m+1 \leq j \leq m+1$) erfüllt sind ($\langle op \rangle$: $\geq$, $>$, $\leq$, $<$).

Ein Lösungsalgorithmus [Hammer & Rudeanu 68] für derartige Systeme läuft in 3 Schritten ab.

Schritt 1:

Bringe jede Ungleichung $g_i(x_1, x_2, \dots, x_n) > d$ in die Form $g_i(x_1, x_2, \dots, x_n) \geq d + 1$, und jede Ungleichung $g_j(x_1, x_2, \dots, x_n) < d$ in die Form $-g_j(x_1, x_2, \dots, x_n) \geq -d+1$. Hier wird vorausgesetzt, daß alle Koeffizienten von $g_i, g_j \in \mathbb{Q}$ sind, was bei unseren Zuordnungsproblemen zutrifft.

Schritt 2:

Bringe jede Gleichung und Ungleichung in die kanonische Form (*) bzw. (2*), d.h.

$$f_i(x_1, x_2, \dots, x_n) = c_{i1} x_{i1}^{a_{i1}} + c_{i2} x_{i2}^{a_{i2}} + \dots + c_{in} x_{in}^{a_{in}} = d_i$$

$$f_j(x_1, x_2, \dots, x_n) = c_{j1} x_{j1}^{a_{j1}} + c_{j2} x_{j2}^{a_{j2}} + \dots + c_{jn} x_{jn}^{a_{jn}} \geq d_j$$

$$\text{wobei } c_{i1} \geq c_{i2} \geq \dots \geq c_{in} > 0$$

$$\text{und } c_{j1} \geq c_{j2} \geq \dots \geq c_{jn} > 0$$

$$\text{für } 1 \leq i \leq m \text{ und } m+1 \leq j \leq m+1$$

Das kann, wie bereits erwähnt, durch die Transformationen $\bar{x}_i = 1 - x_i$ bzw. $x_i = 1 - \bar{x}_i$ jederzeit erreicht werden.

Schritt 3:

Nach der Ausführung von Schritt 2 können Zustandsuntersuchungen eigenständig für jede "Zeile" des PB-G/U-Systems stattfinden, ein hochgradig parallelisierbarer Vorgang. Wenn sich hierbei herausstellt, daß eine PB-Gleichung unlösbar ist (Zustände 1,5) oder das bei einer PB-Ungleichung zutrifft (Zustand 3), dann existieren keine Lösungen des gesamten PB-G/U-Systems. Der Lösungsalgorithmus kann beendet werden. Im nachfolgenden Programm (Bild V.1) wird dieser Fall durch die Variable NONSOL=TRUE beschrieben. Tritt diese Bedingung nicht ein, wird als nächstes geprüft, ob eine redundante Ungleichung (Zustand 1) vorliegt (TRIVIEQU=TRUE). Diese kann ohne Bedeutung für das Gesamtsystem entfernt werden (Prozedur DELIEQU). Bei einer sonstigen Determinierung (DET=TRUE) gewisser Gleichungen (Zustände 2,3,6,7) oder Ungleichungen (Zustände 4,5) müssen Variable den Wert 0 oder 1 annehmen. Gibt es durch Mehrfachdeterminierung einen Widerspruch der Form $x_i = 1$ und $x_i = 0$ ($1 \leq i \leq n$), was durch die Programmvariable CONTRA=TRUE angezeigt wird, dann ist das Gesamtsystem unlösbar. Im anderen Fall werden die Determinierungen eingesetzt und dadurch ein neues PB-G/U-System mit weniger Variablen erzeugt. Das tut die Prozedur UPDATE. Anschließend wird das neue System wieder wie oben auf Determiniertheit überprüft, bis schließlich kein solcher Zustand mehr vorliegt oder das System gelöst ist. Im zweiten Fall ist man fertig, im ersten schaut man nach partiellen Determinierungen (Zustände 4 (Gleichungen) bzw. 2 (Ungleichungen)).PDET nimmt in einem solchen Fall den Wert TRUE an, die Prozedur MAXPDET bestimmt die Gleichung bzw. Ungleichung mit maximalem partiellen Determinierungsgrad p, um daraus p+1 neue PB-G/U-Systeme über n-p Boole Variablen zu erzeugen, die unabhängig voneinander auf Lösbarkeit geprüft werden können (Parallelisierbarkeit des Algorithmus).

In heute üblichen Rechenanlagen muß das jedoch sequentiell getan werden. Deswegen bietet sich für die Ablage dieser PB-Systeme ein Keller an, der mit den Operationen PUSH und POP gefüllt bzw. geleert wird. Das $(p+1)$ -te neue PB-System wird wieder auf Determiniertheit untersucht, usw.. Liegt auch irgendwann keine partielle Determinierung vor, dann muß die undeterminierte Fallunterscheidung einer Boole Variablen x_{k_1} ($1 \leq k \leq m+1$)

durchgeführt werden. Der Index k wird aus empirischen Erfahrungen durch die Prozedur MAXUDET so bestimmt, daß c_{k_1} maximal ist.

Eines der beiden neu entstehenden PB-G/U-Systeme (Erzeugung durch die Prozedur UPDATE) wird zwecks späterer (oder paralleler) Untersuchung in den Keller gesteckt, das zweite wiederum auf Determiniertheit untersucht. Damit ist eine Rekursion des Lösungsalgorithmus vollständig beschrieben. Am Ende einer solchen muß immer festgestellt werden, ob das PB-G/U-System schon gelöst ist. Das zeigt die Variable PB-G/U-S an. Sie hat den Wert FALSE, falls noch keine Lösung gefunden ist. Sonst wird die gefundene Lösung ausgegeben, und es folgt eine Prüfung, ob weitere Gleichungs-/Ungleichungssysteme im Keller stecken, deren Lösbarkeit noch nicht entschieden ist. STACK=EMPTY zeigt den leeren Keller an und führt zum Beenden des Algorithmus. In einer vereinfachten Programmschreibweise ist sein Aufbau im Bild V.1 beschrieben.

Eingabe: PB-G/U-System der Form

$$(1 \leq i \leq m) \sum_{k=1}^n c_{i,k} x_{i,k} = d_i$$

$$(m+1 \leq j \leq m+1) \sum_{k=1}^n c_{j,k} x_{j,k} \geq d_j$$

Ausgabe: Lösungen $(x_i)_{i=1}^m$ mit $1 \leq i \leq n$ und Anzahl der Lösungen NSOL

Bedeutung der Variablen und Prozeduren:

- NSOL: TRUE. Eine PB-Gleichung oder -Ungleichung ist nicht lösbar.
- TRIVIEQU: TRUE. Eine Ungleichung ist redundant.
- DET: TRUE. Eine PB-Gleichung oder -Ungleichung ist determiniert.
- PDET: TRUE. Eine PB-Gleichung oder -Ungleichung ist partiell determiniert.
- CONTRA: TRUE. Es gibt eine widersprüchliche Determinierung einer Variablen x_i . Damit ist das gesamte PB-G/U-System unlösbar.
- MAXDET: Prozedur zur Bestimmung der maximalen partiellen Determinierung.
- MAXIDET: Prozedur zur Bestimmung der maximal undeterminierten Gleichung oder Ungleichung.
- STACK: EMPTY. Es gibt keine gespeicherten, noch zu testenden PB-G/U-Systeme mehr.
- PUSH: Prozedur zum Speichern von noch zu testenden PB-G/U-Systemen.
- POP: Prozedur zum Zurückholen von noch zu testenden PB-G/U-Systemen.
- DELIEQU: Prozedur zum Entfernen einer redundanten Ungleichung.
- UPDATE: Prozedur zur Reduktion von determinierten Boole Variablen.
- OUTPUT SOLUTION/SOL: Prozedur zur Ausgabe einer gefundenen Lösung mit der Angabe über die Zahl aller bisher bestimmten Lösungen.
- STATUS: Prozedur zur Bestimmung der Zustände des PB-G/U-Systems. In dieser Prozedur werden die Variablen NSOL, TRIVIEQU, DET, PDET und CONTRA bestimmt.

```

begin
  DETERMINATE:STATUS
  if NONSOL=TRUE then
    begin
      SOL:=0
      READY
      end  $f_i$ 
    if TRIVIEQU=TRUE then
      begin
        DELIEQU
        PROFSOL
        end  $f_i$ 
      if DET=TRUE then
        begin
          if CONTRA=TRUE then
            begin
              SOL:=0
              READY
              end  $f_i$ 
            UPDATE
            PROFSOL
            end  $f_i$ 
          PDETERMINE:if PDET=TRUE then
            begin
              MAXDET
              while  $i \leq \text{MAXDET}$  do
                begin
                  UPDATE
                  PUSH
                  end od
                PROFSOL
                end  $f_i$ 
              UNDETERMINE:MAXIDET
              UPDATE
              PUSH
              UPDATE
            PROFSOL:if PB-G/U-S-TRUE then
              begin
                OUTPUT SOLUTIONS
                SOL = SOL+1
                READY
                end  $f_i$ 
              DETERMINE
              READY:if STACK=EMPTY then END
                else begin
                  POP
                  end  $f_i$ 
                DETERMINE
              END:OUTPUT SOL
            end
          end
        end
      end
    end
  end

```

Der Algorithmus geht auf HAMMER und RUDEANU [Hammer & Rudeanu 68] zurück und liefert alle Lösungen, soweit überhaupt welche existieren in der Form sogenannter Lösungsfamilien. Jede Familie wird beschrieben durch

$$x_i = 0 \text{ bzw. } x_i = 1 \text{ für alle } i \in T \subseteq \{1, 2, \dots, n\}$$

Die nicht festgelegten Boole Variablen sind die Freiheitsgrade der Familie und können beliebige Werte 0 und 1 annehmen, ohne die Lösungsmenge zu verlassen. Damit ist eine triviale Möglichkeit gegeben, lineare Pseudo-Boole Optimierungsaufgaben zu lösen, d.h. einen Boole Vektor $(x_1, x_2, \dots, x_n)$ zu bestimmen, der den PB-Nebenbedingungen genügt und eine lineare PB-Funktion

o.B.d.A. maximiert. Ist $0(x_1, x_2, \dots, x_n) = \sum_{i=1}^n 0_i x_i^{a_i}$ die Zielfunktion, dann nimmt sie innerhalb einer Lösungsfamilie ihr lokales Maximum an, wenn die Freiheitsgrade $x_i^{a_i} = 1$ gesetzt werden, da ohne Einschränkung die 0_i positiv angenommen werden können. Aus den lokalen Maxima bestimmt man leicht das globale. Das Auffinden der Lösungsfamilien und die Optimierung der Zielfunktion verfügen über keine Rückkopplung, weshalb diese Methode verbessert werden kann. Die Zielfunktion kann nämlich als "dynamische" Nebenbedingung in das PB-G/U-System eingefügt werden. Das geht so: wurde ein Boole Vektor $(x_1^*, x_2^*, \dots, x_n^*)$ bestimmt, der den Nebenbedingungen genügt, dann sind nur noch solche $(x_1, x_2, \dots, x_n)$ interessant, die

$$(*) \quad 0(x_1, x_2, \dots, x_n) > 0(x_1^*, x_2^*, \dots, x_n^*)$$

1. Beschleunigungskriterium

erfüllen und deshalb bessere Lösungen sind. Die Ungleichung (*) wird dem PB-G/U-System zugefügt. Vor dem Auffinden der ersten Lösung $(x_1^*, x_2^*, \dots, x_n^*) \in \{0, 1\}^n$ kann ein heuristischer Zuordnungsalgorithmus, etwa auf der Basis von Listenschedules eine Begrenzung der Kosten feststellen und damit eine initiale Kopplung von Nebenbedingungen und Zielfunktionen des PB-G/U-Systems erreichen.

In [Hammer & Rudeanu 68] ist ein weiteres Beschleunigungskriterium empfohlen. Während bei der Lösung linearer PB-G/U-Systeme eine Rekursion in der Reihenfolge determiniert, partiell determiniert, undeterminiert behandelt wurde, ist empirisch begründbar folgende Variation sinnvoll:

Determinierung;
 Undeterminierte Verzweigung der Zielfunktion, falls
 diese im PB-G/U-System noch vorhanden ist;
 partielle Determinierung;
 undeterminierte Verzweigung.
(2. Beschleunigungskriterium)

Ein drittes Beschleunigungskriterium kann wirksam werden, wenn nach einer gefundenen Lösung eine undeterminierte Verzweigung der Zielfunktionsbeschränkung (*) stattfindet. Sind während des Ablaufs des Algorithmus' für eine Indexmenge $H \subseteq \{1, 2, \dots, n\}$ alle Boole Variablen x_i ($i \in H$) bestimmt (also $x_i = 0$ oder $x_i = 1$ für $i \in H$) und findet eine undeterminierte Verzweigung von x_{o_i} der Zielfunktion

$$\sum_{j=1}^{o_n} o_j x_{o_j}^{a_{o_j}}$$

statt, dann entstehen zwei neue PB-G/U-Systeme durch die Determinierungen $x_{o_1}^{a_{o_1}} = 1$ und $x_{o_1}^{a_{o_1}} = 0$. Wenn das erste eine Lösung besitzt, braucht der zweite Zweig u.U. gar nicht untersucht werden. Dann nämlich, wenn die Beziehung

$$o_1 > \sum_{k \in K} o_k$$

(3. Beschleunigungskriterium)

für $K = \{i: 1 \leq i \leq n \wedge i \notin H \wedge x_{0_i}^{a_0} = 0\}$ erfüllt ist. In dieser Situation hat die Zielfunktion durch die Festlegung $x_{0_1}^{a_0} = 0$ gar keine Gelegenheit, ihren Wert aus dem ersten Zweig zu verbessern.

Dieses Beschleunigungskriterium gewinnt an Schärfe, wenn wir die Gesamtlänge eines Ablaufplans minimieren wollen und hierzu das PB-G/U-System von Satz 11 (Kap. III.3) verwenden. Die Zielfunktion heißt dann

$$\sum_{k=\tau_f(a_0)+1}^{\tau_e(a_0)} k \eta_{i_0}^k = \min$$

und kann anfangs als zusätzliche Nebenbedingung folgender Form in das PB-G/U-System eingehen, weil wir Einheitsbearbeitungszeiten voraussetzen.

$$(*) \sum_{k=0}^{\tau_e(a_{i_0}) - \tau_f(a_{i_0}) - 1} (\tau_e(a_{i_0}) - k) \bar{\eta}_{i_0}^k \tau_e(a_{i_0}) - k \geq \sum_{k=\tau_f(a_0)+1}^{\tau_e(a_0)} k$$

Beachte: $\dots > \tau_e(a_{i_0}) - k > \tau_e(a_{i_0}) - (k+1) > \dots > \tau_f(a_{i_0}) + 1$

Findet während des Ablaufs des Lösungsalgorithmus irgendwann eine undeterminierte Verzweigung von (*) bzgl. der Variablen $\bar{\eta}_{i_0}^{\tau_e(a_{i_0}) - k}$ statt, und konnte für $\bar{\eta}_{i_0}^{\tau_e(a_{i_0}) - k} = 1$ eine Lösung bestimmt werden, dann braucht der Zweig $\bar{\eta}_{i_0}^{\tau_e(a_{i_0}) - k} = 0$ nicht

mehr untersucht werden, da die oben definierte Menge K aus genau einem Element $\tau_e(a_{i_0}) - k_0$ besteht, wobei $k_0 > k$ (Ein-

heitsbearbeitungszeit der Aufgabe a_{i_0}) und somit $\tau_e(a_{i_0}) - k >$

$\tau_e(a_{i_0}) - k_0$ trivialerweise erfüllt ist. Der zweite Pfad kann

also nicht zu einer Verbesserung der bereits gefundenen Lösung führen.

Das hat die Konsequenz, daß alle aufgrund einer undeterminierten Verzweigung der Zielfunktion gespeicherten PB-G/U-Systeme nicht mehr untersucht werden müssen, wenn eine Lösung gefunden wurde. Aber auch diejenigen gespeicherten Systeme, die wegen einer partiellen Determinierung oder einer undeterminierten Verzweigung einer sonstigen Gleichung oder Ungleichung noch neue Lösungen beitragen können, reichen dennoch für eine Verbesserung der Zielfunktion nicht aus. Dafür sorgt das modifizierte Vorgehen bei der Lösung von PB-Optimierungsaufgaben, das bei der Lösung von Zuordnungsproblemen über das Empirische hinaus begründbar wird. Solche Verzweigungen finden nämlich erst dann statt, wenn alle Boole Variablen der Zielfunktion determiniert sind. Lösungen dieser Zweige ergeben alle den gleichen Zielfunktionswert. In der Sprache der Zuordnungsprobleme bedeutet das, daß die zugeordneten Ablaufpläne alle dieselben Gesamtlängen besitzen. Unter den Voraussetzungen von Satz 11 (Kap. III.3) ist die erstgefundene Lösung bereits längenminimal, sofern der Algorithmus die beiden letztgenannten Beschleunigungskriterien verwendet (Bild V.2). Zu seiner Beschreibung sind weitere Prozeduren und Variablen nötig. OF=TRUE kennzeichnet das Vorhandensein der Zielfunktionsvariablen im gegenwärtig untersuchten PB-G/U-System und besagt, daß damit undeterminierte Verzweigungen nach Ausführung aller Determinierungen gemacht werden müssen. Daraus entstehen neue Systeme mit dem Prädikat OFBRANCH=TRUE. Nach dem Auffinden einer Lösung brauchen alle gespeicherten PB-G/U-Systeme mit dem Prädikat OFBRANCH=FALSE gemäß den Vorbemerkungen nicht mehr untersucht werden. Alle anderen können ebenfalls übergangen werden, wenn für sie das 3. Beschleunigungskriterium zutrifft. Das wird durch DROP=TRUE gekennzeichnet.

07-03885-1

System muß wegen des 3. Beschleunigungskriteriums nicht untersucht werden.

[illegible]

V.2 Nicht lineare Pseudo-Boole Methoden

In den Sätzen 12 (Kap. IV.1) und 15 (Kap. IV.3) konnten wir das Zuordnungsproblem $(G, B, ex.g?)$ für p -inhomogene und (p_1, p_2) -heterogene Prozessorsysteme unter der Voraussetzung natürlicher Bearbeitungszeiten und Freigabezeitpunkte äquivalent in die Lösbarkeit nicht linearer Pseudo-Boole Gleichungs-/Ungleichungssysteme umformen. In diesem Abschnitt werden wir sehen, wie solche Systeme mit dem Begriff der sogenannten charakteristischen Funktion gelöst werden können. Hierauf aufbauend sind auch nicht lineare PB-Optimierungsaufgaben algorithmisch behandelbar. Eine Lösungsmöglichkeit geht auf HAMMER, ROSENBERG und RUDEANU [Hammer & Rudeanu 68] zurück, die erst Lösungsfamilien zu den Nebenbedingungen bestimmen und damit aus der Zielfunktion mehrere nicht lineare Pseudo-Boole Funktionen ableiten, deren globale Minimierung die Optimierungsaufgabe löst.

Def. R (Kap. V.2): Sei $\Sigma(x_1, x_2, \dots, x_n)$ ein System Pseudo-Boolescher Gleichungen und Ungleichungen. Eine Boole Funktion

$$\phi: \{0,1\}^n \rightarrow \{0,1\}$$

heißt charakteristische Funktion von $\Sigma(x_1, x_2, \dots, x_n)$, wenn die Lösungen der Boole-Gleichung $\phi(x_1, x_2, \dots, x_n) = 1$ mit denen von $\Sigma(x_1, x_2, \dots, x_n)$ übereinstimmen.

Aus den Lösungsfamilien F_h ($1 \leq h \leq p$) von $\Sigma(x_1, x_2, \dots, x_n)$ kann die charakteristische Funktion einfach ermittelt werden. Bezeichnet man die fixierten Variablen von F_h mit $x_{h_1}, x_{h_2}, \dots, x_{h_{m(h)}}$ und nehmen diese die Werte $x_{h_i} = \xi_{h_i}$ ($1 \leq i \leq m(h)$) an, so berechnet sich ϕ zu

$$\phi(x_1, x_2, \dots, x_n) = c_1 \cup c_2 \cup c_3 \cup \dots \cup c_p$$

$$\text{mit} \quad c_h = x_{h_1}^{\varepsilon_{h_1}} x_{h_2}^{\varepsilon_{h_2}} \dots x_{h_m(h)}^{\varepsilon_{h_m(h)}}$$

Für nicht lineare PB-G/U-Systeme $\Sigma(x_1, x_2, \dots, x_n)$ kann die charakteristische Funktion über eine Linearisierung gewonnen werden, die der Einrichtungssatz ermöglicht. Jede Abbildung $f: \{0,1\}^n \rightarrow \mathbb{R}$ kann danach in die Form

$$(*) \quad f(x_1, x_2, \dots, x_n) = \sum_{i=1}^m a_i P_i(x_1, x_2, \dots, x_n)$$

gebracht werden, wobei in den $P_i(x_1, x_2, \dots, x_n)$ ausschließlich Konjunktionen vorkommen:

$$P_i(x_1, x_2, \dots, x_n) = x_{i_1}^{\pi_{i_1}} x_{i_2}^{\pi_{i_2}} \dots x_{i_{k(i)}}^{\pi_{i_{k(i)}}} \quad (1 \leq i \leq m)$$

Nach einer Substitution dieser Terme durch m neue Boole Variable $y_1, y_2, \dots, y_m$ erhält man die lineare PB-Gleichung

$$f'(y_1, y_2, \dots, y_m) = \sum_{i=1}^m a_i y_i,$$

deren charakteristische Funktion $\Psi(y_1, y_2, \dots, y_m)$ über die Lösungsfamilien bestimmt werden kann. Es ist klar, daß sich die charakteristische Funktion $\phi(x_1, x_2, \dots, x_n)$ von f direkt aus Ψ ableiten läßt:

$$(2*) \quad \phi(x_1, x_2, \dots, x_n) = \Psi(x_{i_1}^{\pi_{i_1}} x_{i_2}^{\pi_{i_2}} \dots x_{i_{k(i)}}^{\pi_{i_{k(i)}}} \mid 1 \leq i \leq m)$$

Das Verfahren kann einfach auf Systeme $\Sigma(x_1, x_2, \dots, x_n)$ übertragen werden und erlaubt mit Hilfe der Beziehung (2*) das Bestimmen der charakteristischen Funktion für nicht lineare PB-G/U-Systeme. Aus ihrer disjunktiven Minimalform lassen sich die Lösungsfamilien ablesen.

In allen Sätzen der Vorkapitel entstanden nie ausschließlich nicht lineare Systeme, einige Nebenbedingungen sind stets linear. Deshalb scheint auf der Grundlage der nachfolgenden Eigenschaft charakteristischer Funktionen eine modifizierte Vorgehensweise sinnvoll. Sind $\Sigma_1(x_1, x_2, \dots, x_n)$ und $\Sigma_2(x_1, x_2, \dots, x_n)$ PB-G/U-Systeme mit den charakteristischen Funktionen ϕ_1 und ϕ_2 . Dann lautet die charakteristische Funktion ϕ des zusammengesetzten Systems

$$\phi(x_1, x_2, \dots, x_n) = \phi_1(x_1, x_2, \dots, x_n) \phi_2(x_1, x_2, \dots, x_n)$$

Damit ist man in der Lage, ein gemischt lineares System $\Sigma(x_1, x_2, \dots, x_n)$ in die linearen und ausschließlich nicht linearen Bestandteile $\Sigma_i(x_1, \dots, x_n)$ ($i = 1, 2$) aufzuspalten und die Berechnung der charakteristischen Funktionen getrennt vorzunehmen. Im ersten Teil geht das unmittelbar über die Lösungsfamilien linearer PB-G/U-Systeme, während im nicht linearen System der Umweg über die Linearisierung gegangen werden muß.

Bei PB-Optimierungsproblemen bleibt anschließend die Aufgabe, die Freiheitsgrade der Lösungsfamilien so zu nutzen, daß eine Zielfunktion o.B.d.A. minimiert wird. Sind wiederum für eine Lösungsfamilie F_h ($1 \leq h \leq p$) $x_{h_i} = \xi_{h_i}$ ($1 \leq i \leq m(h)$)

die fixierten Boole Variablen, dann ergibt deren Einsetzen in κ eine neue, nicht notwendig lineare Zielfunktion

$$\kappa_h: \{0, 1\}^{n-m(h)} \rightarrow \mathbb{R},$$

die z.B. mit dem Basisalgorithmus von HAMMER, ROSENBERG und RUDEANU [Hammer & Rudeanu 68] minimiert werden kann. Dieser Algorithmus wird abschließend vorgestellt und liefert nach Anwendung auf alle κ_h ($1 \leq h \leq p$) ein globales Minimum von κ , das die Nebenbedingungen einhält.

Grundlage beim Bestimmen des Minimums einer nicht notwendig linearen PB-Funktion $f_1: \{0,1\}^n \rightarrow \mathbb{R}$ ist die komponentenweise Linearität [Hammer & Rudeanu 68: S. 113/121]. Es gilt insbesondere:

$$f_1(x_1, x_2, \dots, x_n) = x_1 g_1(x_2, x_3, \dots, x_n) + h_1(x_2, x_3, \dots, x_n)$$

$$\text{für geeignete } g_1, h_1: \{0,1\}^{n-1} \rightarrow \mathbb{R}$$

Satz 17 (Kap. V.2)

Der Vektor $(x_1^*, x_2^*, \dots, x_n^*) \in \{0,1\}^n$ ist minimierender Punkt der Funktion f_1 genau dann, wenn es

- (i) ein Element $p_1^* \in \{0,1\}$ gibt, so daß $x_1^* = \varphi_1(p_1^*, x_2^*, \dots, x_n^*)$ und
- (ii) der Vektor $(x_2^*, x_3^*, \dots, x_n^*) \in \{0,1\}^{n-1}$ minimierender Punkt der Funktion $f_2: \{0,1\}^{n-1} \rightarrow \mathbb{R}$ ist.

Dabei gilt:

$$\varphi_1(p_1, x_2, \dots, x_n) = \begin{cases} 1, & \text{falls } g_1(x_2, x_3, \dots, x_n) < 0 \\ 0, & \text{falls } g_1(x_2, x_3, \dots, x_n) = 0 \\ p_1, & \text{falls } g_1(x_2, x_3, \dots, x_n) > 0 \end{cases}$$

und

$$f_2(x_2, x_3, \dots, x_n) = f_1(\varphi_1'(x_2, x_3, \dots, x_n), x_2, x_3, \dots, x_n)$$

$$\text{mit } \varphi_1'(x_2, x_3, \dots, x_n) = \begin{cases} 1, & \text{falls } g_1(x_2, x_3, \dots, x_n) < 0 \\ 0, & \text{falls } g_1(x_2, x_3, \dots, x_n) \geq 0 \end{cases}$$

Bew.:

siehe [Hammer & Rudenau 68: S. 121]

Da φ_1 charakteristische Funktion zu $g_1(x_2, x_3, \dots, x_n) < 0$ ist, kann sie mit den bereits erwähnten Methoden bestimmt werden. Dabei muß u.U. eine Linearisierung stattfinden. Ein Ersetzen der Boole Operationen $\wedge, \vee, \neg$ durch die anfangs erwähnten äquivalenten Operationen in R läßt sie als Pseudo-Boole Funktion erscheinen, die durch Substitution von x_1 in f_1 eine neue PB-Funktion f_2 mit $n-1$ Variablen definiert, auf die obiges Verfahren erneut anwendbar ist, usw.. Schließlich erhält man

$$f_n = x_n g_n + h_n$$

mit reellen Koeffizienten g_n und h_n . φ_n' erfüllt dann

$$\varphi_n' = \begin{cases} 1, & \text{falls } g_n < 0 \\ 0, & \text{falls } g_n \geq 0 \end{cases}$$

Ein Satz von HAMMER und RUDEANU besagt [Hammer & Rudeanu 68], daß damit der minimale Funktionswert von f_1 bestimmt ist, und zwar zu

$$f_{1\min} = f_n(\varphi_n')$$

Der zugehörige minimierte Vektor ergibt sich aus der einfachen Rekursion

$$\begin{aligned} x_n^* &= \varphi_n' \\ x_{n-1}^* &= \varphi_{n-1}'(x_n^*) \\ x_{n-2}^* &= \varphi_{n-2}'(x_{n-1}^*, x_n^*) \\ &\vdots \\ x_1^* &= \varphi_1'(x_2^*, x_3^*, \dots, x_n^*) \end{aligned}$$

Damit haben wir alles, was benötigt wird, um auch nicht lineare PB-G/U-Systeme mit und ohne Zielfunktionen zu lösen. Das Vor-
gehen zerfällt grob in zwei Teile:

1. Teil: Es werden alle Lösungsfamilien der im allgemeinen nicht linearen Nebenbedingungen bestimmt;
2. Teil: Es wird zu jeder Familie F_h ein minimierender Vektor der allgemeinen nicht linearen, reduzierten Zielfunktion κ_h und sein κ_h -Wert berechnet. Hieraus bestimmt sich das globale Minimum. Das ist eine Lösung der gestellten PB-Aufgabe.

Ein entsprechendes Programm benötigt zahlreiche Prozeduraufrufe: Im einzelnen sind das nacheinander:

PART($\Sigma, \Sigma_1, \Sigma_2$): Aufteilen eines PB-G/U-Systems in lineare und ausschließlich nicht lineare Gleichungen und Ungleichungen.

DETCHARFU(Σ, ϕ): Bestimmung der charakteristischen Funktion für lineare PB-G/U-Systeme. Hier geht der vollständige Algorithmus von Kap. V.1 ein.

LINEARIZE(Σ, Σ'): Linearisieren eines nicht linearen PB-G/U-Systems.

DELINEARIZE(Σ, ϕ, ϕ'): Delinearisieren einer charakteristischen Funktion.

DMF(ϕ): Bestimmung der disjunktiven Minimalform einer Boole-Funktion.

FAMILY(ϕ): Bestimmung der Lösungsfamilien einer DMF.

GEN(κ_h): Erzeugen der reduzierten Zielfunktion κ_h für eine Lösungsfamilie F_h .

MINNLN(κ_h): Minimieren einer nicht linearen PB-Funktion.

MIN($\underline{x}_h$ ($1 \leq h \leq p$): Bestimmen des globalen Minimums aus den lokalen Minima von F_h .

V.3 Beispiele für Zuordnungsprobleme, die mit Pseudo-Boole Methoden gelöst wurden

Im letzten Abschnitt dieser Arbeit werden Pseudo-Boole Methoden auf mehrere Zuordnungsprobleme angewandt. Es wird sowohl die Frage der Existenz von zulässigen Ablaufplänen untersucht, als auch die Bestimmung eines κ -optimalen Ablaufs an mehreren Beispielen durchgeführt. Um die Wirksamkeit der Pseudo-Boole Methoden bei Zuordnungsproblemen beurteilen zu können, werden in einem Fall alle zulässigen Abläufe bestimmt. Sämtliche Berechnungen wurden mit den beiden Algorithmen von Bild V.1 und V.2 gemacht, die zu diesem Zweck in der Programmiersprache FORTRAN auf einer Rechenanlage des Typs CDC CYBER 172 implementiert wurden. Bei allen Beispielen konnten Antworten auf die Fragen $(G,B,ex.\sigma?)$ und (G,B,opt_{κ}) gefunden werden, im äußersten Fall dauerte das 163 Sekunden, wobei diese Angabe nicht sehr aussagekräftig ist. Sie wird stark von der Realisierung des Keller-Mechanismus beeinflusst, der bei sequentiell arbeitenden Rechnern (sequentiell aus der Sicht eines Programms) benötigt wird und hier durch PUSH- und POP-Operationen auf den Hintergrundspeicher realisiert wurde. Das mußte aus Speicherplatzgründen geschehen. Es führt wie die zahlreichen Kontrolldrucke beider Programmvarianten zu hohen Transferraten, die sich in der Rechenzeit nicht vernachlässigbar niederschlagen. Wesentlich aussagekräftiger ist die Anzahl der stattfindenden Rekursionen beim Ablauf der Algorithmen. Aufgrund ihrer hohen Parallelisierbarkeit ist das ein realistisches Komplexitätsmaß, das nicht zuletzt wegen der extrem geringen Anzahl von undeterminierten Rekursionen in allen betrachteten Beispielen gut ausfällt. Ob das Zufall ist oder für Zuordnungsprobleme bestimmter Klassen allgemein gilt, wurde bisher nicht überprüft. Hier bietet sich ein guter Ansatz für Beschränkungsaussagen über die Komplexität. Schließlich sind es die undeterminierten Rekursionen, die dazu führen, daß bzgl. einer Boole Variablen x_i

die beiden Möglichkeiten $x_i=0$ und $x_i=1$ untersucht werden müssen. Bei Determinierungen und partiellen Determinierungen mit $p > 1$ werden sofort erfolglose Pfade im binären Baum erkannt, der durch die Boole Variablen des PB-G/U-Systems aufgespannt wird. In allen gerechneten Beispielen ist das die überwältigende Mehrheit der überhaupt vorhandenen Pfade. Beurteilt man abzählende Algorithmen danach, wieviele Möglichkeiten aller vorhandenen sie untersuchen, um ein endliches Problem zu lösen, dann erweisen sich die betrachteten Pseudo-Boole Methoden als sehr effektiv. Insbesondere ist es damit möglich, Lösungen zu bisher nicht beantwortbaren Zuordnungsproblemen zu finden.

Die gerechneten Beispiele werden nun vorgestellt und Ergebnisse zusammengefaßt. In einer Schlußübersicht sind alle Daten festgehalten, die für eine Beurteilung der Algorithmen wichtig sind.

Beispiel 1: In einem einfachen homogenen Arbeitensystem sollte die generelle Einsetzbarkeit von linearen Pseudo-Boole Methoden überprüft werden. Alle zulässigen Ablaufpläne wurden per Programm ermittelt.

Arbeitsgraph: Bild V.3

Bearbeitungssystem: 1 Prozessortyp mit 2 Prozessoren

Aufgabe	Rechenzeit	Betriebsmittel- forderungen	Freigabe- zeit	Beendigungs- zeit
1	1	Ø	Ø	4
2	1	Ø	Ø	4
3	1	Ø	Ø	4
4	1	Ø	Ø	4
5	1	Ø	Ø	4

Ergebnisse: Es werden 20 Boole Variablen benötigt und 101 Pfade untersucht, um die 13 Lösungen zu bestimmen. Dabei finden statt:

Determinierungen	: 260
partielle Determinierungen	: 93
nichtdeterminierte Rekursionen	: 0
Rechenzeit (in sec)	: 16



Bild V.3: Arbeitengraph zu Beispiel 1 und 2

Beispiel 2: Für den einfachen Arbeitengraph des Bildes V.3 wird mit Pseudo-Boole Methoden ein längenminimaler Ablauf bestimmt.

Arbeitengraph: Bild V.3

Bearbeitungssystem: 1 Prozessortyp mit 2 Prozessoren

Aufgabe	Rechenzeit	Betriebsmittel- forderungen	Freigabe- zeit	Beendigungs- zeit
1	1	0	0	4
2	1	0	0	4
3	1	0	0	4
4	1	0	0	4
5	1	0	0	4

Ergebnisse: Es wurden 20 Boole Variablen benötigt und 8 Pfade durchsucht, ehe der optimale Ablauf gefunden war. Dabei finden statt:

Determinierungen	: 38
partielle Determinierungen	: 4
undeterminierte Rekursionen	: 2
Rechenzeit (in sec)	: 2

Beispiel 3: Für ein 2-inhomogenes Arbeitensystem wird ein längenminimale Ablauf bestimmt. Die Existenz eines zulässigen Ablaufplans ist unmittelbar erkennbar.

Arbeitengraph: Bild V.4

Bearbeitungssystem: 2 Prozessortypen; 2 Exemplare vom Typ 1 und 1 Prozessor vom Typ 2 sind vorhanden.

Aufgabe	Rechenzeit		Betriebsmittel- forderungen	Freigabe- zeit	Beendigungs- zeit
	Typ 1	Typ 2			
1	∞	1	$\emptyset$	$\emptyset$	6
2	∞	1	$\emptyset$	$\emptyset$	6
3	1	∞	$\emptyset$	$\emptyset$	6
4	∞	1	$\emptyset$	$\emptyset$	6
5	1	∞	$\emptyset$	$\emptyset$	6
6	1	∞	$\emptyset$	$\emptyset$	6
7	1	∞	$\emptyset$	$\emptyset$	6
8	1	∞	$\emptyset$	$\emptyset$	6

Ergebnisse: Es werden 24 Boole Variablen benötigt und 26 Pfade durchsucht, ehe ein optimaler Ablauf gefunden ist. Er sieht so aus:

	5	3	8
	6	7	

Prozessortyp 1

1	2	4	
---	---	---	--

Prozessortyp 2

Dabei finden statt:

Determinierungen	: 88
partielle Determinierungen	: 19
undeterminierte Rekursionen	: 2
Rechenzeit (in sec)	: 6

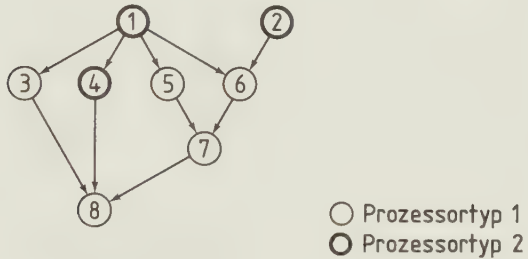


Bild V.4: Arbeitsgraph zum Beispiel 3

Beispiel 4: Es wird die Existenz eines zulässigen Ablaufs für ein 2-inhomogenes Arbeitensystem geprüft, das mit bekannten Algorithmen nicht lösbar ist.

Arbeitsgraph: Bild V.5

Bearbeitungssystem: 2 Prozessortypen mit je 2 Prozessoren

Aufgabe	Rechenzeit		Betriebsmittel-	Freigabe-	Beendigungs-
	Typ 1	Typ 2	forderungen	zeit	zeit
1	∞	1	$\emptyset$	$\emptyset$	3
2	1	∞	$\emptyset$	$\emptyset$	1
3	∞	1	$\emptyset$	$\emptyset$	3
4	∞	1	$\emptyset$	$\emptyset$	4
5	1	∞	$\emptyset$	$\emptyset$	5
6	∞	1	$\emptyset$	$\emptyset$	6
7	1	∞	$\emptyset$	$\emptyset$	1
8	1	∞	$\emptyset$	$\emptyset$	3
9	∞	1	$\emptyset$	$\emptyset$	4
10	1	∞	$\emptyset$	$\emptyset$	5
11	∞	1	$\emptyset$	$\emptyset$	5
12	1	∞	$\emptyset$	$\emptyset$	5
13	∞	1	$\emptyset$	$\emptyset$	1
14	1	∞	$\emptyset$	$\emptyset$	2
15	∞	1	$\emptyset$	$\emptyset$	3
16	∞	1	$\emptyset$	$\emptyset$	4
17	1	∞	$\emptyset$	$\emptyset$	4
18	∞	1	$\emptyset$	$\emptyset$	4
19	1	∞	$\emptyset$	$\emptyset$	5
20	∞	1	$\emptyset$	$\emptyset$	4

Ergebnisse: Es werden 38 Boole Variablen benötigt. Im 323. durchsuchten Pfad steht definitiv fest, daß kein zulässiger Ablauf existiert. Dabei finden statt:

Determinierungen : 966
 partielle Determinierungen: 368
 undeterminierte Rekursion : 1

Rechenzeit (in sec) : 87

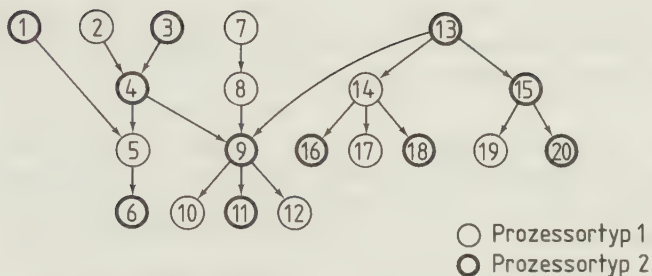


Bild V.5: Arbeitsgraph für Beispiel 4

In den folgenden Beispielen wurde herauszufinden versucht, weshalb im letzten Beispiel kein zulässiger Ablaufplan bestimmt werden konnte.

Beispiel 5: Im Arbeitsgraph des Bildes V.5 werden die Vorgängerbeziehungen zwischen den Aufgaben 4 und 9 und zwischen den Aufgaben 13 und 9 aufgehoben. Dadurch entstehen drei voneinander unabhängige Teilgraphen. Existiert nun ein zulässiger Ablauf?

Arbeitsgraph: Bild V.5 ohne die Kanten (4,9) und (13,9)

Bearbeitungssystem: wie in Beispiel 4

Ergebnisse: Es werden 38 Boole Variablen benötigt. Im 444. durchsuchten Pfad wird die Nichtexistenz eines zulässigen Ablaufs festgestellt. Dabei finden statt:

Determinierungen	: 1582
partielle Determinierungen	: 526
undeterminierte Rekursion	: 1

Rechenzeit (in sec)	: 120
---------------------	-------

Beispiel 6: Nachdem die Lockerung der Vorgängerrelation nicht zur Existenz eines zulässigen Ablaufs führte, wird nun die Anzahl der Prozessoren erhöht.

Arbeitengraph: Bild V.5

Bearbeitungssystem: 2 Prozessortypen mit je 3 Prozessoren; ansonsten wie in Beispiel 4

Ergebnisse: Es werden 32 Boole Variablen benötigt. Im 19. Pfad wurde folgender zulässiger Ablauf gefunden.

2	8	5		10	Prozessortyp 1
7	14	17		12	
		19			
13	4	20	6	11	Prozessortyp 2
1	15	16	9		
3		18			

Dabei finden statt:

Determinierungen	: 44
partielle Determinierungen	: 32
undeterminierte Rekursionen	: 3
Rechenzeit (in sec)	: 7

Beispiel 7: Es stellt sich die Frage nach der Längenminimalität des eben bestimmten Ablaufs (Beispiel 6).

Arbeitengraph: Bild V.5

Bearbeitungssystem: wie im Beispiel 6

Ergebnisse: Um einen längenminimalen Ablauf zu bestimmen, werden 35 Boole Variablen benötigt. Im 5. Pfad ist er gefunden, im 23. durchsuchten Pfad als Optimum bestätigt.

Er hat folgendes Aussehen:

2	14	5	10
7	8	17	12
		19	

Prozessortyp 1

1	4	9	6
3	15	18	11
13		16	20

Prozessortyp 2

(Der Ablaufplan von Beispiel 6 war also nicht optimal)
Dabei fanden statt:

Determinierungen : 76
partielle Determinierungen : 26
undeterminierte Rekursionen: 2

Rechenzeit (in sec) : 7

Beispiel 8: Bisher konnte die Nichtexistenz von Ablaufplänen festgestellt werden, wenn von jedem Prozessortyp nur zwei Exemplare vorhanden sind und die Existenz, wenn je drei Prozessoren zur Verfügung stehen. Wie sieht es aus, wenn vom Typ 1 nur 2, vom Typ 2 jedoch 3 Prozessoren Aufgaben bearbeiten können?

Arbeitengraph: Bild V.5

Bearbeitungssystem: 2 Prozessortypen; Typ 1 mit 2 Exemplaren Typ 2 mit 3 Prozessoren, ansonsten wie Beispiel 4.

Ergebnisse: Aus dem Ablaufplan von Beispiel 8 erkennt man sofort, daß unter der Einschränkung zweier Prozessoren vom Typ 1 ebenfalls ein zulässiger Ablauf existieren muß.

Deshalb wurde sofort ein lngenminimaler bestimmt. Es waren 35 Boole Variablen notig. Der 125. Pfad besttigte die optimale Lsung, die bereits im 10. Pfad gefunden war.

2	14	17	5	10
7	8	19	12	

Prozessortyp 1

1	4	9	11	6
3	15	20	16	
13			18	

Prozessortyp 2

Dabei fanden statt:

Determinierungen : 525
 partielle Determinierungen : 135
 undeterminierte Rekursionen : 2

Rechenzeit (in sec) : 38

Beispiel 9: Wie sieht es mit der Existenz von zulssigen Ablufen aus, wenn umgekehrt vom Typ 1 3 Prozessoren und vom Typ 2 2 Prozessoren vorhanden sind?

Arbeitengraph: Bild V.5

Bearbeitungssystem: 2 Prozessortypen; Typ 1 mit 3 Exemplaren, Typ 2 mit 2 Prozessoren; sonst wie Beispiel 4.

Ergebnisse: Es werden 38 Boole Variablen bentigt. Im 12. durchsuchten Pfad ist die Nichtexistenz eines zulssigen Ablaufs festgestellt. Es finden statt:

Determinierungen : 101
 Partielle Determinierungen : 0
 undeterminierte Rekursionen : 12

Rechenzeit (in sec) : 5

Ergebnisse: Es werden 39 Boole Variablen benötigt. Im 49. durchsuchten Pfad wird eine ebenfalls optimale Lösung gefunden.

2	14	5	10	12	
7	8	17	19		

Prozessortyp 1

13	1	9	10	11	6
3	4	15	18	20	

Prozessortyp 2

Dabei finden statt:

Determinierungen	: 280
partielle Determinierungen	: 87
undeterminierte Rekursion	: 1
Rechenzeit (in sec)	: 18

Beispiel 12: Der späteste Beendigungszeitpunkt der Aufgabe 9 wird um eine Zeiteinheit erhöht.

Arbeitsgraph: Bild V.5

Bearbeitungssystem: wie im Beispiel 4 mit der Änderung, daß nun die Aufgabe 9 statt zum Zeitpunkt 4 erst zum Zeitpunkt 5 fertig sein muß.

Ergebnisse: Es kann kein zulässiger Ablauf existieren, da ein Nachfolger der Arbeit 9, nämlich die Aufgabe 11, als späteste Beendigungszeitpunkt ebenfalls 5 hat. Das wird durch den Lösungsalgorithmus mit 39 Boole Variablen im 310. Pfad festgestellt. Dabei finden statt:

Determinierungen	: 991
partielle Determinierungen	: 289
undeterminierte Rekursion	: 1
Rechenzeit (in sec)	: 87

Beispiel 10: Nun wird versucht, welche spätesten Beendigungszeitpunkte abgeschwächt werden müssen, um mit je zwei Prozessoren der beiden vorhandenen Typen zulässige Abläufe zu erhalten.

Arbeitsgraph: Bild V.5

Bearbeitungssystem: wie in Beispiel 4 mit der Änderung, daß nun die Aufgabe 16 statt zum Zeitpunkt 4 erst zum Zeitpunkt 5 fertig sein muß.

Ergebnisse: Es werden 39 Boole Variablen benötigt. Der 55. Pfad liefert den zulässigen Ablauf.

2	14	5	10	12	
7	8	17	19		

Prozessortyp 1

13	1	9	18	11	6
3	4	15	20	16	

Prozessortyp 2

Er ist längenminimal, da 11 Aufgaben auf 2 Prozessoren frühestens nach 6 Zeitintervallen fertig sein können. Es finden statt:

Determinierungen : 368
 partielle Determinierungen : 102
 undeterminierte Rekursion : 1

Rechenzeit (in sec) 21

Beispiel 11: Der späteste Beendigungszeitpunkt der Aufgabe 20 wird um eine Einheit erhöht.

Arbeitsgraph: wie in Beispiel 4 mit der Änderung, daß nun die Aufgabe 20 statt zum Zeitpunkt 4 erst zum Zeitpunkt 5 fertig sein muß.

Beispiel 13: Die spätesten Beendigungszeitpunkte der Aufgaben 9 und 11 werden um eine Zeiteinheit erhöht.

Arbeitengraph: Bild V.5

Bearbeitungssystem: wie in Bild 4 mit den Änderungen, daß die Aufgabe 9 statt zum Zeitpunkt 4 erst zum Zeitpunkt 5 fertig sein muß und die Aufgabe 11 statt zum Zeitpunkt 5 erst zum Zeitpunkt 6.

Ergebnisse: Es werden 40 Boole Variablen benötigt. Im 575. durchsuchten Pfad steht die Nichtexistenz eines zulässigen Ablaufs fest. Es finden statt:

Determinierungen	:	2183
partielle Determinierungen	:	500
undeterminierte Rekursion	:	1
Rechenzeit (in sec)	:	163

Beispiel 14: Nun werden auch noch die spätesten Beendigungszeitpunkte der Aufgaben 10 und 12 um eine Einheit erhöht.

Arbeitengraph: Bild V.5

Bearbeitungssystem: wie im Beispiel 4 mit folgenden Änderungen:

Aufgabe 9: spätester Beendigungszeitpunkt 5

Aufgabe 11: spätester Beendigungszeitpunkt 6

Aufgabe 10: spätester Beendigungszeitpunkt 6

Aufgabe 12: spätester Beendigungszeitpunkt 6

Ergebnisse: Es werden 41 Boole Variablen benötigt. Im 296. durchsuchten Pfad wird ein zulässiger Ablauf gefunden.

2	14	5			10
7	8	17	19		

Prozessortyp 1

13	1	15	18	6	11
3	4	16	20	9	

Prozessortyp 2

Dabei finden statt:

Determinierungen	:	1787
partielle Determinierungen	:	276
undeterminierte Rekursion	:	1

Rechenzeit (in sec)	:	91
---------------------	---	----

Eine Zusammenfassung der Ergebnisse der Beispiele 10 bis 14 ergibt folgendes Bild. Falls der späteste Beendigungszeitpunkt einer der Aufgaben 16,20 oder 9 um eine Einheit verlängert wird, wobei das auch für Nachfolger dieser Arbeiten gilt, die nur eine Einheit später fertig werden dürfen, reichen je zwei Prozessoren von jedem Typ für die Existenz eines zulässigen Ablaufs aus.

Eine Zusammenfassung der Ergebnisse aller Beispiele ist in der nachfolgenden Übersicht V.6 gegeben. Vorher werden Ansätze für eine Weiterführung dieser Arbeit aufgezeigt.

V.4 Offene Punkte

In dieser Arbeit werden die Zuordnungsprobleme $(G, B, ex.g?)$ und (G, B, opt_n) für eine breite Klasse von Arbeitensystemen mit Pseudo-Boole Methoden behandelt und dadurch lösbar. Im wesentlichen mußten zwei Einschränkungen gemacht werden: die Forderung nach natürlichen Bearbeitungszeiten und natürlichen Freigabezeitpunkten. Dabei erweist sich die zweite Voraussetzung als direkte Konsequenz der ersten. Wegen der Digitalisierung heutiger Rechenanlagen scheint eine Erweiterung beider Bedingungen auf reelle Werte allenfalls von theoretischem Interesse, so daß eine derartige Anwendungsmöglichkeit Pseudo-Boolescher Methoden gering eingeschätzt werden muß.

Frage- stellung	Antwort (ex. σ ?)	Anzahl benötigter Boole Variablen	durchsuchte Pfade	Gesamtzahl der Rekursionen	Determinierungen	Partielle Determinierungen	unterminierte Rekursionen	Rechenzeit (CYBER 172)
Beispiel 1	alle σ	20	101	353	260	93	0	16 sec
Beispiel 2	opt _K	20	8	44	38	4	2	2 sec
Beispiel 3	opt _K	24	26	109	88	19	2	6 sec
Beispiel 4	ex. σ ?	38	323	1333	966	368	1	87 sec
Beispiel 5	ex. σ ?	38	444	2109	1582	526	1	120 sec
Beispiel 6	ex. σ ?	32	19	79	44	32	3	7 sec
Beispiel 7	opt _K	35	23	104	76	26	2	7 sec
Beispiel 8	opt _K	35	125	662	525	135	2	38 sec
Beispiel 9	ex. σ ?	38	12	113	101	0	12	5 sec
Beispiel 10	ex. σ ?	39	55	471	368	102	1	21 sec
Beispiel 11	ex. σ ?	39	49	368	280	87	1	18 sec
Beispiel 12	ex. σ ?	39	310	1281	991	289	1	87 sec
Beispiel 13	ex. σ ?	40	575	2684	2183	500	1	163 sec
Beispiel 14	ex. σ ?	41	296	2064	1787	276	1	91 sec

Bild V.6 Übersicht aller gerechneten Beispiele

Anders verhält es sich mit Effizienzaussagen dieser Techniken. Die Beispiele zeigen durchwegs ein erfreuliches und ermutigendes Bild. Jedoch fehlen allgemeine Komplexitätsmaße, die nötig sind, um die Güte der Pseudo-Boole Methoden bei Zuordnungsproblemen umfassender zu verstehen. Eine weitere Fortführung dieser Arbeit ist in der breiten Verwendbarkeit dieser Methoden bei Zuordnungsproblemen begründet, die es wahrscheinlich macht, für bisher ungelöste Zuordnungsprobleme neue Lösungsalgorithmen abzuleiten. Schließlich weisen die PB-G/U-Systeme, die Zuordnungsproblemen äquivalent sind, eine erstaunliche Symmetrie auf, die es erlauben könnte, neue wirksamere Lösungsalgorithmen für solche Systeme zu bestimmen und damit effizienter zu gestalten. Zusammenfassend scheint eine Fortführung dieser Arbeit in drei Richtungen sinnvoll:

1. Ableiten von Komplexitätsabschätzungen der Pseudo-Boole Methoden bei bestimmten Klassen von Zuordnungsproblemen,
2. Ableiten neuer Zuordnungsalgorithmen (Schedule-Algorithmen) mit Hilfe der Pseudo-Boole Lösungsalgorithmen,
3. Effizienzsteigerungen der Pseudo-Boole Methoden bei PB-G/U-Systemen, die Zuordnungsproblemen äquivalent sind.

Literatur:

- [Adolphson 77] Adolphson D.: Single Machine Job Sequencing with Precedence Constraints, SIAM J. Comp. 6, 40-54 (1977)
- [Adolphson & Hu 73] Adolphson D., Hu T.C.: Optimal Linear Ordering, SIAM J. Appl. Math. 25, 403-424 (1973)
- [Baker & Schrage 76] Baker K.R., Schrage L.E.: Finding an Optimal Sequence by Dynamic Programming: an Extension to Precedence-Related Tasks, Operations Res., to appear (1976)
- [Baker & Su 74] Baker K.R., Su Z.-S.: Sequencing with Due-Dates and Early Start Times to Minimize Maximum Tardiness, Naval Res. Logist.Quar. 21, 171-176 (1974)
- [Bratley et al. 75] Bratley P., Florian M., Robillard P.: Scheduling with Earliest Start and Due Date Constraints on Multiple Machines, Naval Res. Logist.Quart. 22, 165-173 (1975)
- [Brucker et al. 77] Brucker P., Garey M.R., Johnson D.S.: Scheduling Equal-Length Tasks Under Treelike Precedence Constraints to Minimize Maximum Lateness, Mathematics of Operations Research, Vol. 2, No. 3, August 1977
- [Bruno et al. 74] Bruno J., Coffman E.G., Sethi R.: Scheduling Independent Tasks to Reduce Mean Finishing Time, Comm. ACM 17, 382-387 (1974)
- [Coffman & Denning 73] Coffmann, Denning: Operating Systems Theory, Prentice Hall, Inc., Englewood Cliffs, N.J. 1973
- [Dijkstra 76] Dijkstra, E.W.: A Discipline of Programming, Prentice-Hall, 1976

- [Ecker 77] Ecker: Theorie deterministischer Schedules, Bibliographisches Institut Mannheim, 1977
- [Fisher 76] Fisher M.L.: A Dual Algorithm for the One-Machine Scheduling Problem, Math. Programming 11, 229-251 (1976)
- [Fortet 60] Fortet, R.: Applications de l'algèbre de Boole en recherche opérationnelle, Rev. Francaise Recherche Opér. 17-26, (1960)
- [Garey & Johnson 75] Garey, Johnson: Complexity Results for Multiprocessor Scheduling under Resource Constraints, SIAM J. Computing 4, 397-411 (1975)
- [Garey & Johnson 76] Garey M.R., Johnson D.S.: Scheduling Tasks with Nonuniform Deadlines on Two Processors, J. Assoc. Comput. Mach. 23, 461-467 (1976)
- [Gelders & Kleindorfer 74] Gelders L., Kleindorfer P.R.: Coordinating Aggregate and Detailed Scheduling Decisions in the One-Machine Job Shop: part I. Theory, Operations Res. 22, 44-60 (1974)
- [Gelders & Kleindorfer 75] Gelders L., Kleindorfer P.R.: Coordinating Aggregate and Detailed Scheduling Decisions in the One-Machine Job Shop: II-Computation and Structure, Operations Res. 23, 312-324 (1975)
- [Gonzalez 71] Gonzalez M.J.: A Decentralized Parallel Processor System, Dissertation, The University of Texas at Austin December 1971
- [Graham 66] Graham R.L.: Bounds for Certain Multiprocessing Anomalies, Bell System Tech. J. 45, 1563-1581 (1966)

- [Graham 69] Graham R.L.: Bounds on Multiprocessing Timing Anomalies, SIAM J. Appl. Math. 17, 263-369 (1969)
- [Graham et al. 77] Graham, Lawler, Lengstra, Rinnooy Kan: Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey, to be published in: P.L. Hammer, E.L. Johnson, P. Korte: Annals of Discrete Mathematics (Discrete Optimization) Bd. 415, North Holland
- [Hammer & Rudeanu 68] P.L. Hammer (Ivanescu), Sergiu Rudeanu: Boolean Methods in Operations Research and Related Areas, Springer Verlag Berlin-Heidelberg-New York 1968
- [Heller 78] Heller, D.: A Survey of Parallel Algorithmus in Numerical Linear Algebra, SIAM Reviews 10/78 p. 740
- [Horn 72] Horn W.A.: Single Machine Job Sequencing with Treelike Precedence Ordering and Linear Delay Penalties, SIAM J. Appl. Math. 23, 189-202 (1972)
- [Jackson 55] Jackson, J.R.: Scheduling a Production Line to minimize Maximum Tardiness, Research Report 43, Management Science Research Project, University of California, Los Angeles (1955)
- [Karp 72] Karp R.M.: On the Computational Complexity of Combinatorial Problems, in R.E. Miller, J.W. Thatcher (eds) (1972) Complexity of Computer Computations, Plenum Press, New York, 85-103 (1972)
- [Knuth 73] Knuth D.: Private Communication to T.C. Hu, July 73 (1973)
- [Kuck et al. 74] Kuck, Budnik, Chen, Lawrie, Towle, Strebenadt: Measurements of Parallelism in Ordinary FORTRAN Programs, Computer, Jan. 1974, IEEE Computer Society

- [Lageweg et al. 76] Lageweg B.J., Lengstra J.K., Rinnooy Kan A.H.G., Minimizing Maximum Lateness on One-Machine: Computational Experience and Some Applications, Statistica Neerlandica 30, 25-41 (1976)
- [Lawler 73] Lawler E.L.: Optimal Sequencing of a Single Machine Subject to Precedence Constraints, Management Sci. 19, 546-655 (1973)
- [Lawler 75] Lawler E.L.: A "Pseudopolynomial" Algorithm For Sequencing Jobs to Minimize Total Tardiness, Ann. Discrete Math. 1, to appear (1975)
- [Lawler 76] Lawler E.L.: Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints, Ann. Discrete Math., to appear
- [Lawler & Moore 69] Lawler E.L., Moore J.M.: A Functional Equation And Its Application to Resource Allocation and Sequencing Problems, Management Sci 16, 77-84
- [Lengstra -] Lengstra J.K.: Unpublished
- [Lengstra et al. 75] Lengstra J.K., Rinnooy Kan A.H.G., Brucker P.: Complexity of Machine Scheduling Problems, Ann. Discrete Math. 1, to appear (1975)
- [Lüdtke 77] Lüdtke: Die Komplexität von Scheduling-Algorithmen für M-Prozessor Systeme, $M > 2$, TU Berlin FB 20, Dissertation
- [Mahon & Florian 75] McMahon G.B., Florian M.: On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness, Operations Res. 23, 475-482 (1975)

- [Moore 68] Moore J.M.: A N Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs, Management Sci.15, 102-109 (1968)
- [Ramamoorthy et al. 72] Ramamoorthy C.Y., Chandy K.M., Gonzalez M.J.: Optimal Scheduling Strategies in a Multiprocessor System, IEEE Transactions on Computers, Vol. C-21, No. 2, February 1972
- [Rinnooy et al. 75] Rinnooy Kan A.H.G., Lageweg B.J., Lenstra J.K.: Minimizing Total Costs in One-Machine Scheduling, Operations Res. 23, 908-927 (1975)
- [Rothkopf 66] Rothkopf M.H.: Scheduling Independent Tasks on Parallel Processors, Management Sci. 12, 437-447
- [Schindler et al. 74] Hennings, Schindler, Steinacker: The Complexity of Preemptive Scheduling Algorithms for Multiprocessor Systems, TU Berlin FB 20, Bericht 74-20
- [Schindler & Simonsmeier 74] Schindler, Simonsmeier: Scheduling Independent Tasks on Different Processors, TU Berlin FB 20, Bericht 74-05
- [Schindler 75] Schindler S.: Scheduling Algorithms for Monitor Systems on M-Processor Systems, $M \geq 1$, TU Berlin FB 20, (1975)
- [Schmid 78] Schmid H.J.: A Geometrical Interpretation of the Hungarian Method, Discrete Mathematics 21, 297-308 (1978)
- [Sidney 73] Sidney J.B.: An Extension of Moor's Due Date Algorithm, in: s.E. Elmaghraby (ed) (1973) Symposium on the Theory of Scheduling and its Applications. Lecture Notes in Economics and Mathematical Systems 86, Springer Berlin, 393-398

- [Sidney 75] Sidney J.B.: Decomposition Algorithms for Single-Machine Sequencing with Precedence Relations and Deferral Costs, Operations Res. 23, 283-298 (1975)
- [Smith 56] Smith W.E.: Various Optimizers for Single-Stage Production, Naval Res. Logist. Quart. 3, 59-66 (1956)
- [Steinacker 78] Steinacker M.: Korrektheitsbeweise bei Scheduling-Problemen, GI - 8. Jahrestagung (1978), Kurzvorträge S. 69
- [Stern 76] Stern H.I.: Minimizing Makespan for Independent Jobs on Nonidentical Parallel Machines - an Optimal Procedure, Working Paper 2/75, Department of Industrial Engineering and Management, Ben-Gurion University of the Negor, Beer-Shera (1976)
- [Valiant 78] Valiant: The Complexity of Combinatorial Computation: An Introduction, GI - 8. Jahrestagung (1978) Tagungsband 326-337

Lebenslauf

Am 23.2.1948 wurde ich, Hartmut Wunderatsch, in Bärenbrunn Kreis Münchberg (Bayern), als Sohn der Eheleute Fritz und Edith Wunderatsch, geb. Eckardt, geboren. Nach Besuch der Grundschule, abgelegter Reifeprüfung am Naturwissenschaftlichen Gymnasium in Münchberg im Jahre 1967 und abgeleistetem Grundwehrdienst bei der Bundeswehr studierte ich von Mai 1969 bis Mai 1973 Mathematik an der Universität Erlangen-Nürnberg. Den Studienabschluß bildete die Diplomprüfung in Mathematik. Seit Juli 1973 bin ich wissenschaftlicher Mitarbeiter am Institut für Mathematische Maschinen und Datenverarbeitung der Universität Erlangen-Nürnberg.

Erlangen, im Februar 1979

In letzter Zeit sind folgende Arbeitsberichte des Instituts
für Mathematische Maschinen und Datenverarbeitung erschienen :
=====

Band 10
=====

- NR. 1 Hoffmann, W. :
Modelle und Analysen von Teilnehmerrechnungssystemen,
angewendet auf das Teilnehmersystems RESPOND unter
CD 3300 MASTER.
(Februar 1977)
- NR. 2 Körber, K., Werzinger, G. :
Das Modell SIM-LAB.
Simulation eines klinisch-chemischen Laboratoriums.
(März 1977)
- NR. 3 Altmann, W., Weber, D. :
Programmierungsmethodik.
(März 1977)
- NR. 4 Maehle, E. :
Fehlertolerante Rechnerstrukturen.
(April 1977)
- NR. 5 Beth, T. :
Kombinatorik: Theorie der Vertretersysteme.
(April 1977)
- NR. 6 Mertens, P. :
Einige Hinweise zur technischen Gestaltung von
Bildschirmdialogen.
(Mai 1977)
- NR. 7 Jäpel, D. :
Ein Begriffssystem zur Formulierung von Schutzaspekten
und seine Anwendung auf Programmsysteme.
(Juni 1977)
- NR. 8 Brendel, W., Grötsch, E., Nagl, M., Schneider, H.-J. :
Graphersetzungssysteme, Beiträge zur Theorie, Anwen-
dung und Implementierung.
(Juni 1977)
- NR. 9 Rüdell, W. :
Darstellung und Vergleich abstrakter Modelle zur
Beschreibung geschützter Systeme.
(Juli 1977)
- NR. 10 Bolch, G., Gall, R., Grassl, M., Schmidt, B. :
Modelle für Trommelspeicher.
(Juli 1977)
- NR. 11 v. Eichhorn, K. :
Steiner Systeme, Codes und affine Räume.
(Juli 1977)

Band 10 (Forts.)

=====

- NR. 12 Götthler, H. :
Zweistufige Graphmanipulationssysteme für die
Semantik von Programmiersprachen.
(Juli 1977)
- NR. 13 Keramidis, S. :
Ein allgemeines formales Modell für Betriebs-
systeme und dessen Implementierung.
(September 1977)
- NR. 14 Arnold, F., Benker, L., Grützt, M. :
Die Fast Fourier Transformation und Möglich-
keiten ihrer Implementierung auf EGPA.
(November 1977)
- NR. 15 Gardill, R., Händler, W., Hessling, H.,
Klar, R., Spies, P. P. :
ERES
Eine nichtprozedurale Rechnerentwurfssprache
mit präziser Zeitbeschreibung.
(November 1977)
- NR. 16 Wichmann, H. :
Entwicklung und Bewertung von Algorithmen für
ein mikroprogrammiertes Parallel- und Assoziativ-
konzept auf Mikroprogramm- und Benutzerebene.
(Dezember 1977)
- NR. 17 Brüggmann, H. W. :
EBIDIS - Strukturierte Bildprogrammierung im Dialog.
(Dezember 1977)
- NR. 18 Jahresbericht 1977 der Informatik.
(März 1978)

Band 11
=====

- NR. 1 Körber, K., Werzinger, G. :
Das Modell SIM-LAB.
Beschreibung des Grundmodells.
(Februar 1978)
- NR. 2 Körber, K., Werzinger, G. :
Das Modell SIM-LAB.
Beschreibung der Modellerweiterungen und der Strategien.
(Februar 1978)
- NR. 3 Körber, K., Werzinger, G. :
Das Modell SIM-LAB.
Listing der Quellprogramme.
(Februar 1978)
- NR. 4 Beth, Th. :
On Resolutions of Steiner Systems.
(März 1978)
- NR. 5 Leeb, K. :
Salami-Taktik beim Quader-Packen.
Riess, W. :
Zwei Optimierungsprobleme auf Ordnungen.
(April 1978)
- NR. 6 Händler, W., Henning, W., Kleinöder, W., Kneissl, J.,
Volkert, J. :
EPIK - Erlanger Projekt Interaktive Kartographie.
(April 1978)
- NR. 7 Schreiber, H. :
Hardware-Messung und Analyse des Ablaufgeschehens
in Rechnerkernen.
(April 1978)
- NR. 8 Weber, D. :
Datengraphen und deren Transformation :
Ein Konzept zur Spezifikation von Datentypen.
(Juni 1978)
- NR. 9 Musielak, H., Schmidt, B., Stössel, M. :
Vergleich von Simulationssprachen :
GPSS-FORTRAN - GPSS - GASP IV.
(Juli 1978)
- NR. 10 Solymosi, A. :
Synthese von analysierenden Automaten auf Grund
von formalen Grammatiken.
(Juli 1978)
- NR. 11 Meyer, K., Schaller, K., Schmidt, B. :
SIM-QUEUE.
Beschreibung eines Simulationsmodells für ein
komplexes Warteschlangensystem, das mit Hilfe von
Eingabedaten zusammengestellt werden kann.
(Juli 1978)

Band 11 (Forts.)

=====

- NR. 12 Bodendorf, F., Eckardt, T., Mertens, P., Schrammel, D. :
Benutzung und Beurteilung von Kleinrechnersystemen in
einer Hochschule - Ein Beitrag zur Dezentralisierungs-
diskussion.
(August 1978)
- NR. 13 Rabinovitz, M. :
An Approach to a Reconfigurable Multiprocessor System.
Bode, A., Händler, W. :
Classification d'architectures parallèles:
Introduction de la notation ECS et application au
projet EGPA.
Sigmund, W. :
Parallel Machine Functions in Program Transformation
and Execution.
(September 1978)
- NR. 14 Beth, Th., Strehl, V. :
Materialien zur Codierungstheorie.
(September 1978)
- NR. 15 Wendler, K. :
Betriebssystemaspekte in hierarchisch modularen
Polyprozessorsystemen - Modellierungsansätze und
Koordinierungsmechanismen.
(Oktober 1978)
- NR. 16 Altmann, W. :
Beschreibung von Programmoduln zum Entwurf
zuverlässiger Softwaresysteme.
(Oktober 1978)
- NR. 17 Hoffmann, W. :
Warteschlangenmodelle für die Parallelver-
arbeitung.
(November 1978)
- NR. 18 Stöth, G. :
Höhere Programmiersprachen für Kleinrechner :
Anforderungen, Entwicklungskriterien, Compiler-
aspekte.
(November 1978)
- NR. 19 Bode, A., Händler, W. :
Rechnerarchitektur - Grundlagen und Verfahren.
(November 1978)
- NR. 20 Kneissl, F. :
Grundsoftware für die Textverarbeitung.
(Dezember 1978)
- NR. 21 Leidel, W. :
Synchronisationsprobleme in Automatenetzen.
(Dezember 1978)

Band 11 (Forts.)

=====

- NR. 22 Bodendorf, F., Renninger, W., Schaller, K. :
 Zwei neue Methodenbanksysteme für statistische
 Anwendungen.
 (Dezember 1978)
- NR. 23 Jahresbericht 1978 der Informatik.
 (März 1979)

Band 12

=====

NR. 1 Elzer, P. :
Strukturierte Beschreibung von Prozesssystemen.
(Februar 1979)

-

